UNIVERSIDAD CARLOS III DE MADRID

Trabajo Fin de Grado
"Simulación de MIMO-OFDM en el downlink de LTE"

GRADO EN INGENIERÍA DE SISTEMAS DE COMUNICACIONES

DEPARTAMENTO DE TEORÍA DE LA SEÑAL Y COMUNICACIONES

Autor: Emilio López Álvaro
Tutora: Dra. Ana García Armada

Septiembre de 2013

RESUMEN

El crecimiento de las telecomunicaciones en estos últimos años ha sido realmente vertiginoso. Los usuarios cada vez exigen una mayor movilidad y un mayor ancho de banda, además de la mejor calidad de servicio posible y una mayor cobertura. Pretenden disponer de cualquier tipo de aplicación en cualquier lugar y en cualquier momento.

Esta tendencia implica una enorme fijación en la actualidad por las redes de comunicaciones inalámbricas de alta capacidad.

Recientemente en España y en otros países del mundo se ha producido el lanzamiento de la red LTE, lo que ha supuesto una enorme evolución con respecto a los sistemas móviles anteriores. Esta nueva tecnología constituye el camino hacia las comunicaciones 4G, sin llegar a cumplir los requisitos de velocidad y tasa de datos de la misma, por lo que se considerará como una tecnología 3.9G. La evolución de este estándar constituirá LTE-Advanced y en este caso sí que se cumplirían las condiciones necesarias fijadas por la Unión Internacional de Telecomunicaciones para ser calificado como 4G.

LTE (Long Term Evolution) rompe con el diseño de sistemas anteriores donde era posible el transporte de voz y datos. En este caso únicamente es posible la transmisión y la recepción de datos ya que la red se basa completamente en el protocolo IP.

Este trabajo fin de grado se basa en la simulación del enlace descendente de LTE mediante el uso de Matlab. La Release 8 de 3GPP define MIMO-OFDM como el método para implementar este DL. Los moduladores a utilizar según el estándar son el QPSK, el 16-QAM y el 64-QAM.

Esta combinación de las tecnologías MIMO y OFDM resulta altamente beneficiosa. OFDM supone una mayor defensa contra los errores surgidos debido a los canales multitrayecto. Por otro lado MIMO consiste en el uso de varias antenas, lo que implica un enorme aumento de la velocidad de transmisión y proporciona al sistema una alta robustez.

OFDM (Orthogonal Frequency Division Multiplexing), divide el ancho de banda en portadoras ortogonales entre sí, esta ortogonalidad implica que se puedan poner bastantes subportadoras juntas, lo que aumenta la eficiencia espectral.

La rejilla de recursos servirá para organizar de forma eficiente los recursos radio para el envío de la información y la introducción de señales piloto. Estas señales de referencia tendrán la función de indicar las variaciones originadas por los efectos del canal y ayudar a que el ecualizador las corrija.

Por otro lado para evitar la interferencia intersimbólica ISI, se usará un prefijo cíclico. Esto supone que el sistema sea robusto frente a los canales multitrayecto.

La implementación del modelo de bloques de OFDM en Matlab será básica para el funcionamiento de todos los sistemas del presente proyecto independientemente del número de antenas y del tipo de modulador a utilizar.

MIMO (Multiple Input Multiple Output) consiste en el uso de varias antenas en el transmisor y en el receptor. Fundamentalmente el incremento del número de antenas y de los flujos de información conlleva una mejora en el rendimiento, debido al aumento de la capacidad del sistema y la cancelación de interferencias.

Con cada antena del sistema existirá una rejilla de recursos diferente. Cada una de estas rejillas dispondrá de unas posiciones para medir cuales son los cambios producidos por el canal, el uso de estas posiciones concretas significa que en el resto de rejillas de recursos esa determinada posición no puede usarse y así el ecualizador calculará los efectos del canal de forma correcta.

Finalmente después de la implementación de todos los sistemas posibles y tras la validación de los mismos, se realizará una valoración de los resultados obtenidos con el objetivo de determinar las características fundamentales de todos ellos y las posibilidades que ofrecen.

ABSTRACT

The growth of the telecommunications has been significantly fast in the last few years. The users demand a greater mobility and more bandwidth as well as the best possible quality of service and better coverage. They want to have any application, in any place, at any time.

Nowadays this trend implies a huge fixation by the high-capacity wireless communications.

Recently in Spain and in other countries in the world, the LTE network launch has occurred which has been a great evolution over earlier mobile systems. This new technology is the way to the fourth-generation communications, without fulfilling the requirements of speed and data rate of the same, and this is the reason why it is considered like a 3.9 G technology. The evolution of this standardization will turn in LTE-Advanced and in this situation, the essential conditions set by International Telecommunications Union to be considered 4G would be achieved.

LTE (Long Term Evolution) has changed the design of previous systems where the communication was with data and voice. In this case, only the transmission and reception of data are possible because the network is completely based on the Internet Protocol.

This final project consists in the simulation of the LTE downlink using Matlab. Release 8 in 3GPP defines MIMO-OFDM as the method to carry this downlink. The modulators for this technology are QPSK, 16-QAM and 64-QAM, according to the standard.

The use of MIMO and OFDM together is really beneficial. OFDM means a greater defense against the multipath channel mistakes. On the other hand, MIMO uses some antennas, the consequences are a great increase of the transmission speed and a system more efficient.

OFDM (Orthogonal Frequency Division Multiplexing) divides the bandwidth into some orthogonal carriers, this orthogonality means that it is possible to use many carriers together in order to improve spectral efficiency.

The resource grid serves to organize the radio resources in an efficient way to send the information and to introduce the pilot signals. The function of the reference signals is to show the changes caused by the channel and to help the equalizer repair it.

A cyclic prefix is used to avoid inter-symbol interference. The use of the cyclic prefix makes the system strong against multipath channels.

MIMO (Multiple Input Multiple Output) entails using some antennas in the transmitter and in the receiver. The increased number of antennas and information flows

mainly involve an improvement in performance, due to the system's capacity gain and the interference cancellation.

There is a different resource grid with each antenna of system. Every one of these grids have some positions to measure what changes have been produced by the channel, the use of these specific positions implies that this resource cannot be used by the other time-frequency grids and, thereby, the equalizer can calculate the channel effects correctly.

Finally, after the implementation of all possible systems and their validation, an evaluation of the results is done in order to define the main features of all of them and the possibilities that they provide.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

El presente trabajo de fin de grado consiste en la simulación de MIMO-OFDM (Multiple Input Multiple Output –Orthogonal Frequency Division Multiplexing) en el enlace descendente de LTE (Long Term Evolution), para el desarrollo del mismo se usará la herramienta Matlab.

El objetivo de este proyecto es estimar el canal mediante el uso de las tecnologías mencionadas en el párrafo anterior y conseguir comprobar las prestaciones de LTE tal y como está definido en el estándar.

1.1 OBJETIVOS

- Profundizar en conceptos de sistemas móviles, teoría de la comunicación y comunicaciones digitales.
- Mejorar y perfeccionar conceptos de la programación en Matlab.
- Seleccionar los estándares de 3GPP adecuados para la elaboración del presente trabajo de fin de grado.
- Conocer los cometidos básicos de los bloques funcionales del sistema OFDM y como realizar su implementación en Matlab.
- Comprender como organizar el flujo de información en la trama radio y en la rejilla de recursos de OFDM.
- Razonar posibles implementaciones para estimar el canal mediante el ecualizador de la mejor forma posible, antes de desarrollar la solución teórica del mismo.
- Adaptar la implementación del sistema OFDM con una única antena transmisora y una única receptora al uso de varias antenas mediante el sistema MIMO.
- Interpretar y comprender los resultados de las gráficas de validación y simulación de los sistemas desarrollados en el proyecto mediante la comparativa entre varios de ellos.
- Deducir las diferentes aplicaciones y usos de las implementaciones de todas las tecnologías realizadas y cuando es recomendable utilizar un sistema u otro.

1.2 ESTRUCTURA DE LA MEMORIA

El presente trabajo fin de grado consiste consta de los siguientes capítulos:

- **Capítulo 1: Introducción**

  En el capítulo introductorio se trata el contexto del presente trabajo fin de grado, los objetivos del mismo, las motivaciones para realizarlo y la estructura de la memoria del proyecto.

- **Capítulo 2: Estado del arte**

  Se explican las tecnologías previas y su evolución hacia LTE.

- **Capítulo 3: LTE**

  Se procede a la explicación de los aspectos más importantes de la tecnología LTE, sus orígenes, sus características, la evolución de LTE en el futuro y el estado de su actual implantación en España y en el resto del mundo.

- **Capítulo 4: OFDM**

  En este capítulo se ilustra esta tecnología y su estrecha relación con el proyecto realizado.

- **Capítulo 5: MIMO**

  Se detalla la tecnología MIMO y las ventajas del uso de múltiples antenas en las comunicaciones móviles. Además hay una descripción del receptor Zero-Forcing y del canal SCM.

- **Capítulo 6: Descripción del sistema desarrollado**

  Esta parte de la memoria consiste en la explicación de cómo se ha desarrollado el trabajo fin de grado con MATLAB y la interpretación de los aspectos teóricos para su correcta realización.

- **Capítulo 7: Validación**

  Se incluyen las mediciones realizadas para comprobar que el sistema está bien implementado y que todo funciona de forma correcta.

- **Capítulo 8: Resultados de simulación**

  Se realizan mediciones de todos los aspectos del sistema desarrollado, con el objetivo de obtener los resultados que demuestren las ventajas del uso de esta tecnología.

- **Capítulo 9: Conclusiones**

  En este apartado se explican las principales ideas extraídas tras la realización del proyecto. Además se describirán las posibles elecciones entre los diferentes tipos de sistemas MIMO-OFDM vistos en el presente trabajo fin de grado, en función de los requerimientos y funciones de la tecnología a implementar.

- **Capítulo 10: Líneas de trabajos futuros**

  Se comentan que posibles cambios futuros podrían desarrollarse en el proyecto.

- **Capítulo 11: Planificación**

  Se detalla la secuenciación de las tareas realizadas.

- **Capítulo 12: Presupuesto**

  Se indican cuáles serían los costes totales del presente trabajo fin de grado.

- **ANEXO:**

  Se incluye el código MATLAB desarrollado. También constará del glosario de términos y las referencias bibliográficas usadas en el trabajo.

1. INTRODUCTION

This final project consists of a MIMO-OFDM (Multiple Input Multiple Output – Orthogonal Frequency Division Multiplexing) simulation of the LTE (Long Term Evolution) downlink. Matlab is the computer program used to develop it.

It is necessary to estimate the channel using the two above-mentioned technologies in order to complete the project correctly. The aim is to achieve the LTE features as they are defined in the 3GPP standard.

1.1. OBJECTIVES

- Explore the concepts of Mobile Systems, Communication Theory and Digital Communications in depth
- Improve Matlab programming
- Select the suitable 3GPP standard in completion of this Bachelor's thesis
- Learn the basic functions of OFDM system blocks and understand their implementation in Matlab
- Understand how to organize the information flow in the radio frame and in the resource grid of OFDM
- Think out possible implementations to estimate the channel by the equalizer in the best way possible, before developing the theoretical solution of the same
- Adjust the OFDM system implementation with a single transmitting antenna and a single receiving to the use of multiple antennas using the MIMO system
- Realize the difference between the results of the validation graphs and of the simulation graphs through the comparison of certain probability curves
- Deduce several applications and uses for all the technologies implementations as well as the preference of use of one over others

1.2. REPORT STRUCTURE

This final project contains the following chapters.

- **Chapter 1: Introduction**

  This introductory chapter is about the context of this Bachelor's thesis, its objectives, the motivations for carrying it out and the project´s report structure.

- **Chapter 2: State of the art**

  There is an analysis of the previous systems and their evolution to LTE.

- **Chapter 3: LTE**

  The most important aspects of LTE technology are explained, as well as its beginnings, its characteristics, LTE future evolution and its current implementation in Spain and in other countries in the world.

- **Chapter 4: OFDM**

  This chapter illustrates this technology and its close connection with this project.

- **Chapter 5: MIMO**

  MIMO technology and the advantages of the use of multiple antennas in mobile communications are explained in detail. Also there is a description about the Zero-Forcing receiver and the SCM Channel.

- **Chapter 6: Description of the developed system**

  This part of the report consists of an explanation of how to implement this project using Matlab and how to interpret theoretical aspects with the aim to complete the work correctly.

- **Chapter 7: Validation**

  The measures to check that the system is performing well and that everything works properly are included.

- **Chapter 8: Simulation results**

  Measures of the entire of the developed system matters are taken with the aim of gaining results that prove the advantages of the use of this technology.

- **Chapter 9: Conclusions**

  The main ideas extracted after carrying out the project are explained. Also, the potential choices to develop the different types of MIMO-OFDM systems that appear in the project are described, according to the requests and the technology functions of implementation.

- **Chapter 10: Future work**

  Future changes that could be developed in this project are discussed.

- **Chapter 11: Planning**

  Sequencing of the tasks performed is described.

- **Chapter 12: Project cost**

  In this final chapter which is prior to the 'Enclosed' one, an estimate of the budget is calculated.

- **ENCLOSED**

  In this chapter, the Matlab code, a glossary of terms and the references used in the project are included.

2. ESTADO DEL ARTE

En esta parte de la memoria, se explicarán las tecnologías móviles existentes antes de LTE, se citarán sus aspectos clave y sus modificaciones hasta llegar al sistema actual desarrollado en este trabajo fin de grado.

Es importante conocer las razones que llevaron a pasar de una generación a otra hasta llegar a la situación actual. Esta evolución de los sistemas tiene como uno de sus objetivos el intento de amortizar la red y no desaprovechar la implementación existente cuando aparece una nueva tecnología. Además es interesante comprobar los continuos cambios producidos y el proceso de avanzar de una conmutación de circuitos a una conmutación de paquetes.

2.1. Sistemas 2G: GSM

GSM (Global System for Mobiles) consiste en la segunda generación de móviles, fue especificado en 1982 y fue introducido comercialmente en 1992.

Permite conexiones a usuarios móviles o fijos con redes tanto fijas como móviles. Su uso es mundial y está dedicado principalmente a la voz mediante una red de circuitos conmutados, de hecho en la actualidad GSM continua siendo la tecnología que más tráfico de voz transporta. A pesar de que GSM sea una tecnología orientada principalmente a la voz también dispone de servicios de datos.

Los servicios de voz ofrecidos por este sistema móvil son una Telefonía FS (Full Speed) a 13 kbps o una HS (Half Speed) a 6,5 kbps, además existe la posibilidad de realizar llamadas de emergencia.

Existen servicios de datos a 9,6 kbps y también es posible el envío de Mensajes Cortos (SMS).

Estos son los servicios fundamentales aunque también existen otras funciones suplementarias que fueron desarrollándose y perfeccionando en diferentes fases, como por ejemplo, el desvío de llamadas o la restricción de las mismas.

Aparte de la prestación de los servicios básicos también hay una serie de funciones de movilidad. Es viable la itinerancia o roaming, que permite a los usuarios conectarse a una red ajena y disponer de las mismas funcionalidades que se encontrarían si estuviesen en su propia red.

GSM proporciona seguridad en sus comunicaciones. Se utilizan un conjunto de números para poder identificar al usuario y al terminal usado. También hay unos mecanismos de autentificación que proporcionan confidencialidad a la identidad del abonado, a los datos de usuario y a los datos de señalización. Además existe un cifrado de la señalización y de la información de usuario cuando esta información se transmite en la interfaz radio con el objetivo de evitar posibles escuchas ilegales.

La modulación usada es la GMSK (Gaussian Minimum Shift Keying) que destaca por su continuidad de fase y su eficiencia espectral.

GSM dispone de un acceso múltiple TDMA/FDD que opera en la banda de frecuencias de 900 MHz, más tarde también en la de 1800 MHz. [1]

2.2. Sistemas 2,5G: GPRS

Los sistemas 2,5 G surgen a partir de la evolución de GSM y constituyen el camino entre las tecnologías 2G y los sistemas 3G.

Existen una serie de modificaciones en las tramas y en los slots de las mismas (ahora se pueden unir slots) que permiten aumentar la velocidad de GSM sin modificar la red. De esta forma se obtiene la tecnología **HSCSD (High Speed Circuit Switched Data)** que consigue alcanzar 14,4 kbps por cada slot.

GPRS (Global Packet Radio Service) implica no desechar la red GSM y realizar cambios sucesivos en la misma. El proceso consiste en dejar la parte de circuitos existente para el tráfico de voz y añadir una red de datos con conmutación de paquetes. La implantación de la red GPRS a la estructura de GSM se inició en el año 2000.

Esta subred nueva, basada en las especificaciones de la Release 97, permite el acceso a redes de datos TCP/IP y su integración con la red móvil. Es necesario realizar en la red una distinción en tiempo real, que no se hacía en GSM, entre el tráfico de voz y datos para trasladar la información a su subred correspondiente.

Las tasas alcanzadas en la práctica con la Release 97, gracias a la posibilidad de unir varios slots de GSM, son de 14 kbps en el enlace de subida y 40 en el descendente. La velocidad teórica que puede alcanzarse con este sistema en futuras mejoras es de 171 kbps.

GPRS no solo aporta una mejora en la velocidad. Aparte de una conexión permanente a las redes de datos mediante los terminales también conlleva una optimización de la compartición de los recursos radio, ya que en la conmutación de paquetes se comparten recursos mientras que en circuitos el recurso es para un único usuario.

Otra novedad es que a partir de esta tecnología la facturación comienza a realizarse en función del tráfico. El coste del servicio deja de estar asociado al tiempo de conexión.

A partir de GPRS con la Release 98 surge **EDGE (Enhanced Data rates for Global Evolution)**. Este GPRS evolucionado se obtiene al aumentar la velocidad por slots mediante una modulación y codificación adaptativas. Se usa la modulación GSMK como en GSM y además la 8-PSK. El modulador 8-PSK de 3 bits por símbolo, implica que se triplique la velocidad por slots pero como desventajas, al disponer de unos símbolos más juntos aumenta la probabilidad de error y las interferencias, lo que

conduce a que sea necesario aumentar la calidad y recibir la información con una buena relación señal a ruido para que el uso de este modulador sea beneficioso.

La velocidad puede alcanzar los 384 kbps, este importante incremento supone que EDGE se quede muy cerca de ser considerado tecnología 3G.

La presente información acerca de las características de los sistemas 2.5 G y los datos de sus velocidades tienen como base la referencia bibliográfica [2].

2.3. Sistemas 3G: UMTS

UMTS (Universal Mobile Telecommunications System) es la tecnología considerada como 3G. Las especificaciones de UMTS realizadas por 3GPP comenzaron con la Release 99, donde se continua con la separación entre circuitos y paquetes siguiendo la disposición de la red GSM/GPRS. Las siguientes versiones fueron la Release 2000, Release 5, Release 6…

A pesar de que se estudió la posibilidad de implementar una red "All-IP" en estas especificaciones esto no se produce hasta desarrollar la tecnología LTE y disponer únicamente de un dominio de paquetes.

En estos estándares se determinaron las características de los servicios portadores para definir los servicios y aplicaciones de los usuarios pretendiendo sentar las bases de un tráfico a alta velocidad.

Como se ha señalado en la introducción de los sistemas 3G, la arquitectura de red de UMTS se basa en la red GSM/GPRS. Uno de los objetivos es disponer del mayor número de elementos comunes entre esta red y la red GSM, además de intentar tener el máximo número de elementos comunes entre el dominio de paquetes y el dominio de circuitos.

El objetivo es incrementar la capacidad, conseguir que el sistema sea más robusto frente a las interferencias y disponer de más usuarios por celda.

Dentro del acceso radio destaca el soft handover de UMTS, es posible trabajar con dos celdas a la vez cuando se está realizando el traspaso, en cambio en GSM existía un hard handover que requería primero una liberación de recursos antes de usar la siguiente celda.

Estos sistemas usan las tecnologías TDMA y WCDMA. WCDMA (Wideband Code Division Multiple Access), usado por el sistema UMTS, se encarga de introducir unos códigos para separar unas señales de otras, de esta forma se podrán enviar varias señales con la misma frecuencia pero con un código diferente. Cada uno de estos códigos se corresponde con un usuario distinto y de esta forma todos los usuarios pueden transmitir de forma simultánea.

La asignación del ancho de banda puede realizarse de forma dinámica, estableciendo a cada usuario el ancho de banda disponible que necesite.

La capacidad de UMTS puede llegar hasta los 2 Mbps cuando existe una baja movilidad.

Más adelante fue desarrollado **HSPA (High-Speed Packet Access)** que mejora la eficiencia espectral de UMTS mediante el uso de nuevos moduladores (16-QAM) entre otros cambios realizados en la trama y en la arquitectura de red.

La Release 5 define HSDPA (High-Speed Downlink Packet Access) que aumenta la velocidad en el enlace de bajada hasta los 14,4 Mbps y en la Release 6 se especifica HSUPA (High-Speed Uplink Packet Access) cuyo enlace de subida llega hasta los 5,72 Mbps.

HSPA+ (HSPA Evolved) es establecido en las Release 7 y 8. Se consigue llegar hasta 42 Mbps en el enlace de bajada y hasta 11 Mbps en el ascendente. La forma de conseguirlo es con una modulación 16-QAM en el enlace de subida y 64-QAM en el de bajada. Además se introduce por primera vez el uso de múltiples antenas para el enlace descendente. [3]

2.4. Actualidad: LTE

LTE (Long Term Evolution) es el sistema móvil más reciente y cuyo uso ya ha comenzado a extenderse por todo el mundo. A pesar de conseguir unas grandes mejoras con respecto a sus predecesores, debido a la exigencia de mayor velocidad y calidad por parte de los usuarios, su diseño no impide una convivencia con las tecnologías previas vistas en los puntos anteriores.

LTE se plantea como la base de las comunicaciones móviles en el futuro, siendo esencial en el proceso de evolución hacia la Cuarta Generación, con unas posibilidades impensables hace unos pocos años.

Después de haber explicado como las comunicaciones móviles han evolucionado hasta el desarrollo de LTE como la tecnología predominante a corto plazo; en el capítulo posterior, se explicarán detenidamente todos los aspectos relacionados con este sistema, sus características más importantes, el impacto que puede suponer en la sociedad actual, el estado de su implantación en España y en el mundo y también las perspectivas de futuro teniendo como base LTE.

3. LTE

3.1. INTRODUCCIÓN

LTE es una tecnología que inicialmente fue pensada como una evolución de UMTS, sin embargo en ese camino hacia los sistemas 4G, se ha convertido en la tecnología fundamental para la evolución de todos los sistemas móviles existentes. [4]

Hubo una enorme batalla entre LTE y WiMAX por ver quien se postulaba como la principal tecnología de banda ancha móvil 4G.

Inicialmente WiMAX, que contaba con el apoyo de Sprint y Cisco, parecía ser la favorita para convertirse en 4G.

Sin embargo Verizon y AT&T apostaron por LTE en 2007 debido a que presenta una evolución más natural a partir de sus predecesoras GSM/UMTS/HSPA/CDMA. Como consecuencia LTE se ha convertido en el referente de la Cuarta Generación.

Una de las principales razones para el desarrollo de esta nueva tecnología fue el hecho de que los usuarios cada vez demandan mayores velocidades para satisfacer sus necesidades y conseguir que nuevos servicios innovadores o aplicaciones como televisión móvil o potentes juegos multijugador online, tengan una buena calidad de servicio. Esta tecnología es hasta 10 veces más rápida que 3G.

Además de estas mejoras para los usuarios, LTE dotará de importantes mejoras económicas a las empresas y los países que adopten esta tecnología. Un estudio en Estados Unidos del operador telefónico EE del Reino Unido tuvo como resultados el aumento en un 67% de la productividad en las empresas, la reducción del 44% en el tiempo invertido y el aumento en un 79% de las relaciones de negocios. También se espera que mejore sectores como la electrónica, la automoción, la economía o la salud, además LTE ofrece herramientas que permitirán a los emprendedores y medianas empresas competir con las grandes empresas. [5]

A largo plazo se busca que en el futuro éste sea un estándar único en las comunicaciones móviles lo que implicará que cualquier terminal fuera de su red de origen pueda acceder a cualquier servicio en cualquier otra red y que exista una mayor competencia en el mercado que beneficie a los usuarios.

En realidad a LTE se le considera una tecnología 3.9G (más allá de 3G y anterior a 4G) debido a que no alcanza los requerimientos suficientes para ser considerado 4G. Por tanto LTE es el referente en el camino hasta alcanzar la Cuarta Generación sin ser realmente parte de la misma.

La tecnología 4G se alcanzará con el sistema LTE-Advanced que consiste en la mejora del estándar LTE. LTE-Advanced posee una serie de mejoras con respecto a LTE que se explicarán con más detalle posteriormente en el apartado 3.4.

3.2. CONCEPTOS FUNDAMENTALES

LTE fue definida en la Release 8 de 3GPP que fue cerrada a finales del año 2008, asegurándose que en el futuro continúe la competitividad establecida en los anteriores sistemas 3G.

La velocidad máxima es de 100 Mbps en el enlace descendente y de 50 Mbps en el ascendente con un ancho de banda de 20 MHz.

La latencia, es decir, el retardo en la respuesta desde la red, es menor de 10 ms.

En 5 MHz pueden actuar hasta 200 usuarios activos.

La flexibilidad de espectro abarca desde 1.4 MHz hasta 20 MHz.

La movilidad está optimizada para 0-15 km/h, tiene buenas prestaciones cuando la velocidad se sitúa entre 15 y 120 Km/h, es posible a 350 Km/h o incluso a 500 Km/h.

A diferencia de 2G o 3G, que fueron diseñados para transportar voz y datos, LTE está optimizado para paquetes datos, es decir, la red se basa en el protocolo IP.

La tecnología de acceso a LTE que se usa en el enlace descendente y que constituye la base principal en la elaboración de este proyecto es MIMO-OFDM que se expondrá más adelante con mayor detalle.

La tecnología MIMO-OFDM busca la combinación del aumento de capacidad y robustez generado por MIMO y la protección frente a los canales multitrayecto desarrollada por OFDM.

A continuación se explicará la evolución de su arquitectura de red, en la que se busca que haya una baja complejidad y se simplifiquen sus elementos. Estos cambios conducen a menores costes de operación de las redes ya que los equipos radio tienen una mayor eficiencia energética.

3.3. ARQUITECTURA DE RED

La arquitectura de red en LTE, como se ha explicado anteriormente, tiene como objetivo fundamental la disminución de la complejidad de sus elementos.

En la red de acceso radio, se reducen elementos y además ahora muchas de sus funciones se realizan de forma conjunta.

En la parte del Core Network, destaca como cambio fundamental la separación del control y los datos. Ambos se separan debido a que el tráfico de control está relacionado con el número de usuarios y las sesiones establecidas, en cambio la parte de datos tiene que ver más con el tipo de aplicaciones que se usen y su complejidad. La información crece más rápido con las aplicaciones que con los usuarios.

A continuación se explicará la arquitectura de red con una figura ilustrativa y con una descripción sencilla de todos los elementos que la componen ([1] [6]).

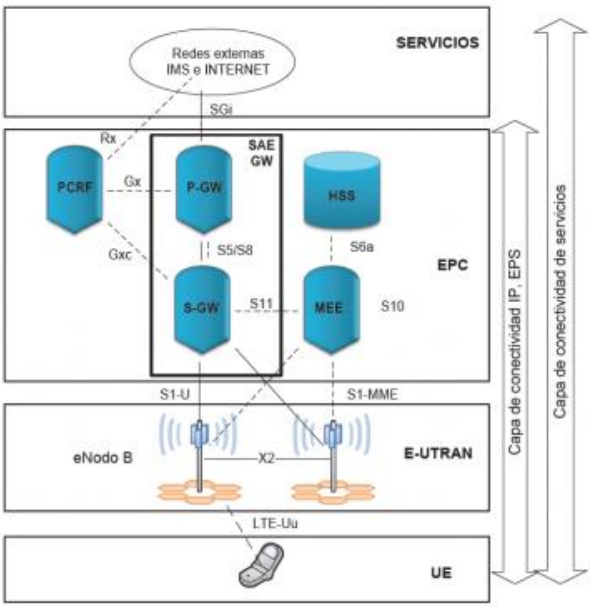

Figura 3.1 *Arquitectura de red LTE*

UE es el terminal móvil.
La interfaz radio LTE-Uu es la encargada de separar el terminal de la eUTRAN.

En la eUTRAN se encuentra el eNodoB. El eNodoB es la estación base de LTE, sus funciones básicas son la gestión de los recursos radio, la sincronización y el control de interferencias, el cifrado y la protección de los datos de usuario, la compresión de cabeceras IP, la selección del MME y el encaminamiento hacia el S-GW.
El eNodoB engloba funciones que en los sistemas 2G o 3G se realizaban de forma independiente por dos elementos diferenciados, el NodoB y RNC constituían la UTRAN en 3G.
La interfaz X2 sirve para comunicar distintos eNodosB y facilitar el soft-handover (traspaso suave). Esta interfaz en anteriores sistemas no existía.
La interfaz S1-MME permite la señalización con el MME.
La interfaz S1-U intercambia el tráfico de usuario con el S-GW.

El MME (Mobile Management Entity) se utiliza para la gestión de la señalización del plano de control de la EPS, el control de seguridad en acceso, la selección del S-GW y P-GW y la itinerancia y la autenticación.
La interfaz S11 tiene como función la señalización entre los usuarios y su correspondiente S-GW.
La interfaz S6a se encarga de la comunicación entre el MME y el HSS.

El S-GW (Serving Gateway) tiene como función fundamental la gestión del tráfico de usuario mediante el encaminamiento de paquetes. Interviene en la calidad del

servicio. Define unos servicios portadores en toda la red y unas calidades asociados a ellos.

En un momento concreto el usuario solo está asociado a una S-GW que en el caso de itinerancia se encuentra separado del P-GW en la red visitada.

La interfaz S5/S8 actúa de pasarela de la red de datos P-GW, S8 para el caso en el que haya itinerancia y S5 cuando no la haya. Cursa el tráfico de usuario y la señalización de movilidad IP.

El P-GW (Packet Data Network Gateway, PDN-GW) realiza la conexión con las redes IP externas a través de la interfaz SGi. Se encarga también de la facturación y del filtrado de paquetes.

El HSS (Home Subscriber Server) es el servidor de subscripción de abonados, almacena los datos de usuarios, los datos de subscripción de los mismos y la lógica de gestión de usuarios. Además tiene información para la movilidad entre LTE y otras redes de acceso.

El PCRF (Policy and Charging Rules Function) controla la facturación por flujo y toma decisiones con respecto a las políticas del control del tráfico.

El EIR (Equipment Identity Register) no aparece en la figura previa. Es la base de datos con la información de los distintos tipos de usuario, indica cuales no pueden acceder a la red (debido a robo o malfuncionamiento). Se comunica con el MME a través de la interfaz S13.

EPC es la evolución hacia 4G del núcleo de la red de las arquitecturas de redes móviles actuales.

Uno de los objetivos fundamentales de la nueva arquitectura de LTE es la convergencia e integración de las redes.

Existe una mayor flexibilidad entre las diferentes redes de acceso y una mayor velocidad en el acceso radio.

Se produce la integración completa con las tecnologías de Internet, todos los servicios ofrecidos se rigen por la conmutación de paquetes.

3.4. LTE-ADVANCE

LTE Advanced es una mejora de LTE estandarizado por 3GPP. Como se ha mencionado antes LTE-Advanced es la primera tecnología 4G aunque comúnmente se considera a LTE como 4G.

Este sistema destaca por su enorme velocidad, llegando a picos de tasas de 1Gbit/s. Esta tecnología sí cumple los requisitos IMT-Advanced definidos por la Unión Internacional de Telecomunicaciones para que una tecnología sea considerada 4G. Entre estos requisitos destaca una tasa de datos de 1Gbps para baja movilidad y una de 100 Mbps para alta movilidad.

La siguiente figura muestra la espectacular evolución de las velocidades de los servicios de comunicaciones móviles en los últimos años.

Impacto de mercado	2003	2006	2009	2010	2016
Tasa máxima	384 kbps	7 Mbps	42 Mbps	~ 150 Mbps	~ 1Gbps
Tasa típica de bajada de usuario	~ 200 kbps	1-2 Mbps	2-10 Mbps	10-20 Mbps	~ 30-100 Mbps
Tasa típica de subida de usuario	64 kbps	64-884 Mbps	0.5-4.5 Mbps	5-10 Mbps	~ 10-60 Mbps

Figura 3.2 *Comparativa velocidades 3G y 4G*

Las mejoras principales de LTE-Advanced con respecto a LTE se centran en las tasas de datos de pico y promedio, en la eficiencia espectral y la latencia en el plano de usuario y de control.

Las exigencias respecto a las tasas de datos se solucionan usando en el enlace descendente hasta MIMO 8x8 y en el ascendente hasta MIMO 4x4, también se realiza la agregación de hasta cinco portadoras de 20 MHz cada una, en espectro contiguo o no contiguo, para obtener un ancho de banda de 100 MHz, aunque en la práctica estará limitado a dos portadoras.

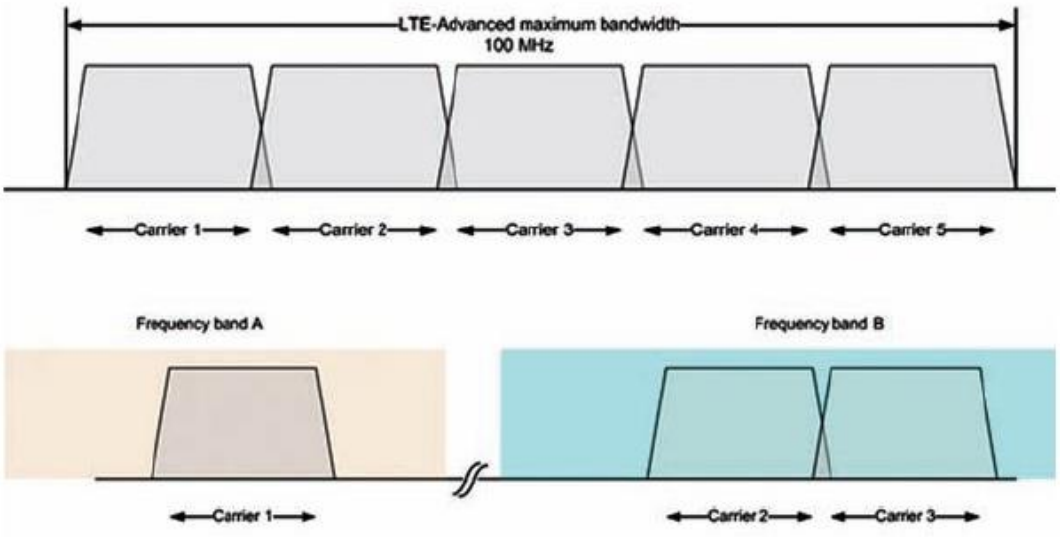

Figura 3.3 *Agrupación de portadoras contiguas y no contiguas en LTE-Advanced [7]*

3.5. IMPLEMENTACIÓN DE LA RED: ESPAÑA Y EL RESTO DEL MUNDO

Para el despliegue actual de la red hay que tener en cuenta que esta tecnología debe coexistir con sus predecesoras y es fundamental que se produzca una correcta interoperabilidad entre las mismas. Actualmente existe la dificultad añadida de que existen muchos tipos de células, la estructuras multicapa combinan macrocélulas, microcéloculas, picocélulas y femtoceldas, lo que complica la situación.

Para que sea posible el uso de esta tecnología aparte de la implementación de la red, es necesario que existan dispositivos móviles compatibles con la red LTE. En los últimos meses se han presentado los primeros terminales que usan esta tecnología entre los que cabe destacar Samsung Galaxy IV y el iPhone V de Apple. Por tanto no todos los smartphones de la actualidad podrían usar esta comunicación.

A finales de Mayo de 2013 Vodafone fue la primera empresa de telecomunicaciones en lanzar su oferta de conectividad LTE en España. Orange, Yoigo y Movistar también lo han hecho recientemente.

El retraso de la implementación en España tiene como principales razones, aparte del problema económico en la situación de crisis actual, el conflicto existente por las bandas del espectro radioeléctrico. Existen tres anchos de banda disponibles para 4G, la banda más codiciada es la de 800 MHz porque ofrece más cobertura y necesita menos inversión, su problema es que sería necesario trasladar la TDT (que actualmente ocupa esa banda) a otra frecuencia, lo que implicaría un proceso de "reantenización"; otra banda posible es la de 1800 MHz que ya está ocupada por los servicios 2G y 3G y la última banda posible es la de 2600 MHz que implicaría un mayor coste al partir la inversión desde cero y además no tiene tanto alcance como el resto. Existen también problemas de planificación al no existir consenso para determinar cuáles serán las redes que se usen en los próximos años. [5]

En el resto del mundo existen 100 millones de conexiones (57 millones de estas conexiones son de Canadá y Estados Unidos) y se espera que se llegue 134 millones a finales de este 2013. Actualmente hay 172 redes LTE comerciales, que se espera que aumenten a 250 para finales de 2013. Existen más de 450 compromisos para desplegar LTE. En 2018 las conexiones LTE llegarán a los 1.000 millones.

4. OFDM

4.1. INTRODUCCIÓN Y ORÍGENES

OFDM (Orthogonal Frequency-Division Multiple) es el método de acceso múltiple de la tecnología LTE en el enlace de bajada. También se usa en el sistema de transmisión inalámbrica de datos WiMAX, en la televisión digital terrestre, en el enlace DSL, en la radio digital DAB o en el sistema de transmisión de datos PLC, entre otros.

El origen de OFDM se remonta a usos militares en la década de los 50 y los 60. Se comenzó a dividir el espectro en múltiples sub-portadoras con el objetivo de resolver los desvanecimientos multitrayecto que se producían en las comunicaciones radio.

Uno de los primeros desafíos fue reducir la separación entre portadoras para evitar el solapamiento. R.W. Chang en 1966 resolvió el problema del solapamiento al conseguir que las portadoras adyacentes fuesen ortogonales entre sí.

En 1971 gracias a Weinstein y Ebert, comenzó a usarse la DFT, cuyo objetivo era agrupar varios números complejos, uno por cada subcanal.

En 1980 Peled y Ruiz propusieron el uso del prefijo cíclico.

Estos conceptos, que se introdujeron a lo largo de los años hasta dar lugar al sistema OFDM actual, serán detallados en el siguiente apartado.

4.2. CONCEPTOS FUNDAMENTALES

Esta tecnología divide el ancho de banda en muchas portadoras, en lugar de enviar la información en un único ancho de banda.

Las subportadoras son ortogonales entre sí, lo que implica que haya una separación mínima entre ellas y aumente la eficiencia espectral.

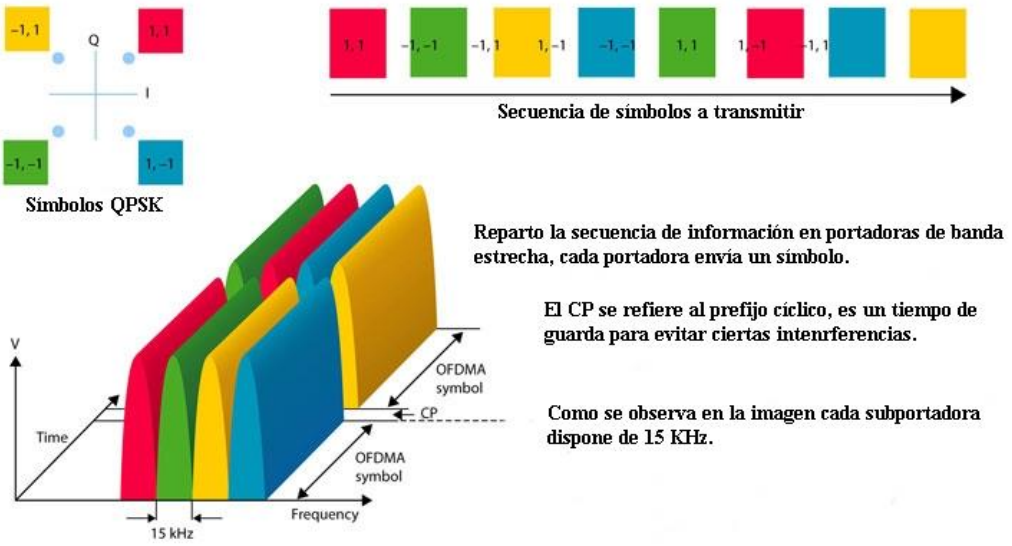

Figura 4.1 *Organización Tiempo-Frecuencia en OFDM*

La ortogonalidad se consigue gracias a que en el momento en el que existe un máximo en una portadora existen nulos en el resto, lo que implica que se puedan poner muchas portadoras juntas, ya que mientras una subportadora está activa las demás están inactivas. [8]

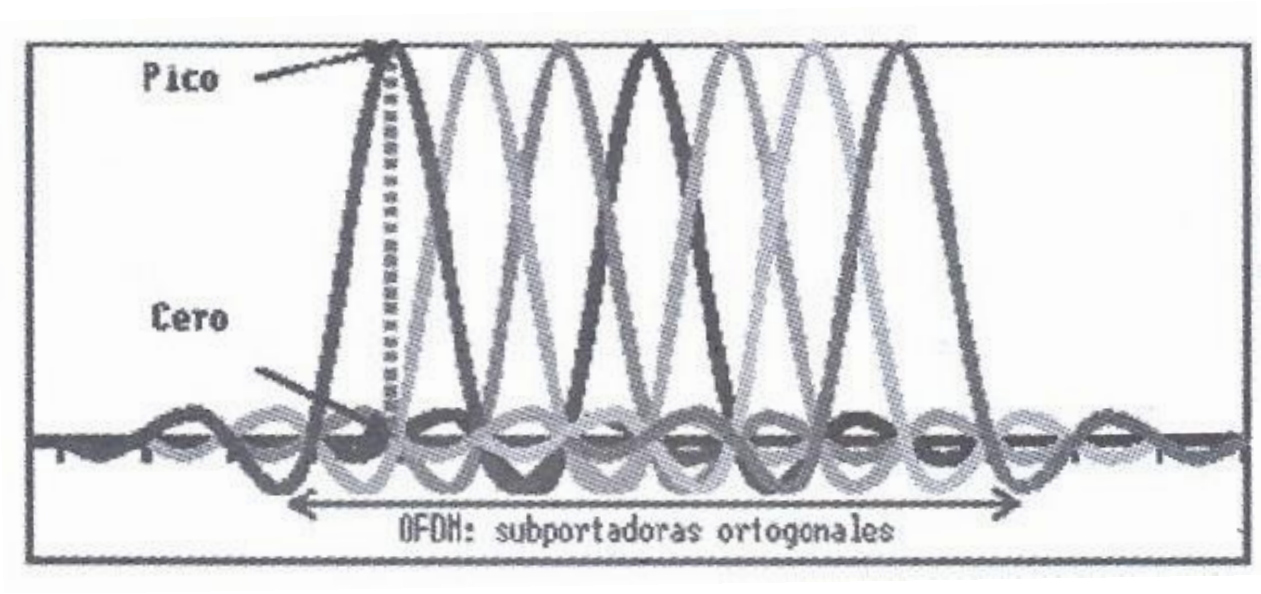

Figura 4.2 *Ortogonalidad de las sub-portadoras*

No todas las frecuencias van a usarse para transmitir información. Algunas de las subportadoras son frecuencias pilotos para tomar distintas referencias. En las portadoras laterales se dejan unas frecuencias de guarda que no se usan.

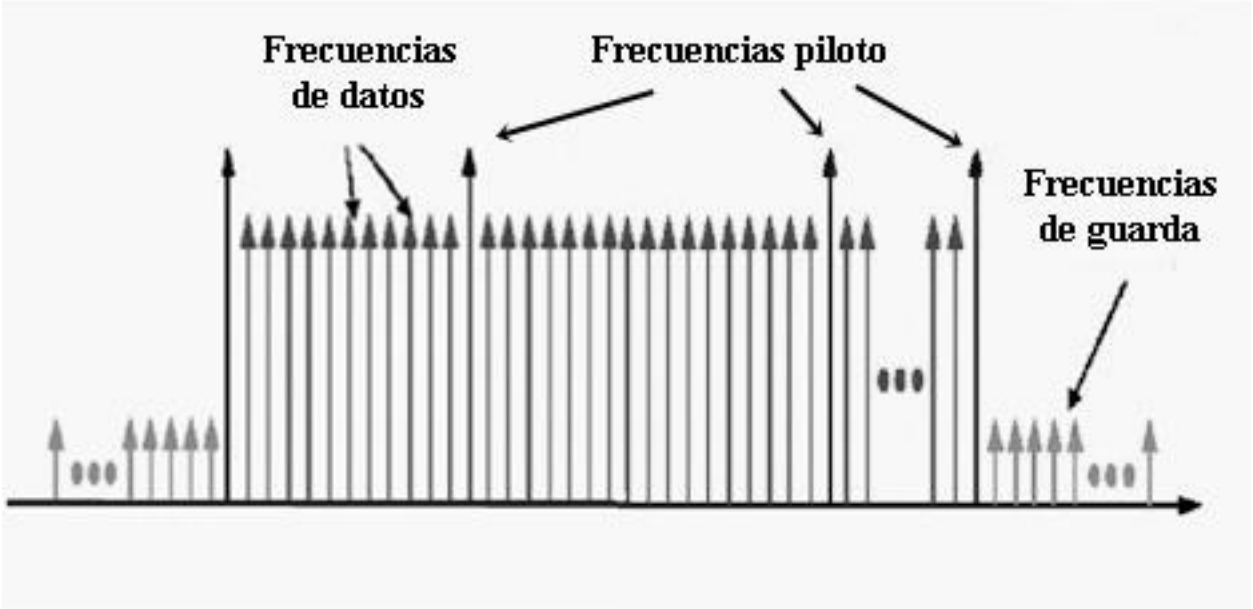

Figura 4.3 *Ubicación de las distintas frecuencias*

El número de frecuencias pilotos y de guarda dependerá del tipo de ancho de banda que necesite el sistema de comunicaciones.

Destaca por su buen rendimiento en canales con desvanecimiento selectivo en frecuencia, además de sus buenas propiedades espectrales y la compatibilidad con los receptores y antenas avanzadas de los últimos tiempos.

Uno de los problemas principales son los medios con multitrayecto (la información de la señal es recibida en distintos instantes de tiempo). Para evitar esto en los sistemas anteriores se añadía un tiempo de guarda. La información de ese intervalo de tiempo no se tiene en cuenta ya que la información puede proceder del multitrayecto. En OFDM se usa un prefijo cíclico como se explicará posteriormente. La adición de este prefijo cíclico implica que el receptor no tenga que lidiar con la interferencia inter-simbólica (ISI). [9]

Sin embargo, el receptor sí que tiene que tratar con el impacto del canal sobre las diferentes sub-portadoras, que experimentan cambios de fase y amplitud. Esta estimación de canal se ve facilitada teniendo símbolos de referencia. Este asunto de los símbolos de referencia o pilotos será tratado posteriormente en el apartado 4.4.

4.3. BLOQUES FUNCIONALES

A continuación se describirán los diferentes bloques funcionales que constituyen el sistema OFDM.

4.3.1. TRANSMISOR

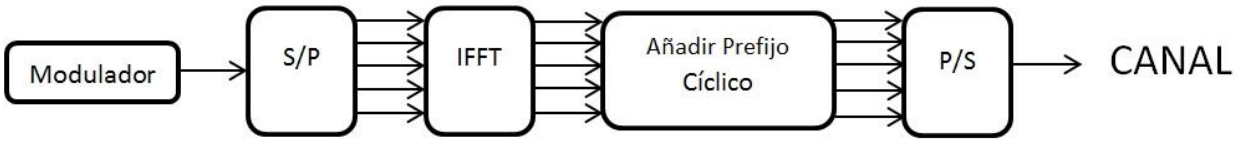

Figura 4.4 *Esquema de bloques del transmisor*

Modulador

Tal y como se estipula en la Release 8, en el enlace descendente de LTE se pueden usar tres tipos distintos de moduladores: QPSK, 16-QAM y 64-QAM, que serán los encargados de convertir los bits de entrada en símbolos complejos.

Estos símbolos complejos serán los encargados de alimentar al conversor serie paralelo y además al bloque de la IFFT.

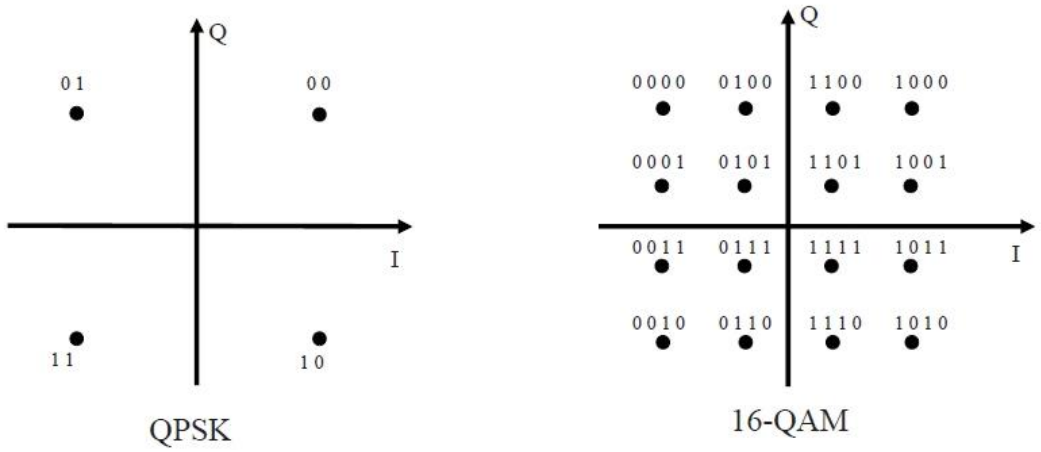

Figura 4.5 *Constelaciones QPSK y 16-QAM*

Conversor Serie Paralelo

El objetivo de este bloque es convertir los símbolos en serie que entran al sistema en una matriz definida por tantas filas como portadoras disponga el sistema, este número vendrá definido por el tamaño de la FFT usada, que dependerá del ancho de banda estipulado en la transmisión. Las columnas resultantes representan el dominio temporal. Cada señal será trasladada a su correspondiente frecuencia portadora que será la que finalmente realizará la transmisión.

Otro aspecto importante de este bloque y que será posteriormente clave en el ecualizador es la adición de diferentes símbolos de referencia intercalados dentro la matriz de forma estratégica según el estándar.

Gracias a la correcta ubicación de estos símbolos, más tarde en el receptor (concretamente en el bloque del ecualizador) se podrán interpolar los efectos del canal para las diferentes sub-portadoras.

El objetivo de estos símbolos es evaluar el comportamiento del canal sobre cada uno de ellos, para poder corregir en el ecualizador la degradación sufrida por los símbolos transmitidos debido al efecto del canal multitrayecto.

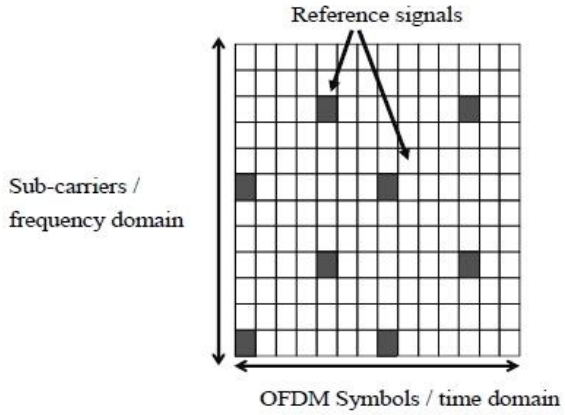

Figura 4.6 *Posición de las señales de referencia o pilotos*

IFFT

La IFFT (Inverse Fast Fourier Transform) se encarga de pasar la señal del dominio de la frecuencia al dominio del tiempo, tanto la IFFT como la FFT son fundamentales en la implementación del sistema OFDM. La matriz obtenida consistirá en la superposición de distintas IFFTs realizadas de forma consecutiva sobre cada columna de la matriz de entrada.

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\omega) e^{j\omega t} d\omega = \mathcal{F}^{-1}\{X(\omega)\}$$

Figura 4.7 *Fórmula de la Transformada de Fourier*

Siendo x(t) la señal continua resultante en el dominio del tiempo y X(ω) la señal en el dominio de la frecuencia.

Añadir Prefijo Cíclico

La razón de añadir una extensión cíclica consiste en evitar la ISI.

El prefijo cíclico consiste en añadir la parte final del símbolo y copiarla al principio del símbolo. Su tamaño debe ser mayor que la respuesta impulsiva del canal.

El efecto de este símbolo previo se evita eliminando posteriormente en el receptor esta extensión cíclica añadida.

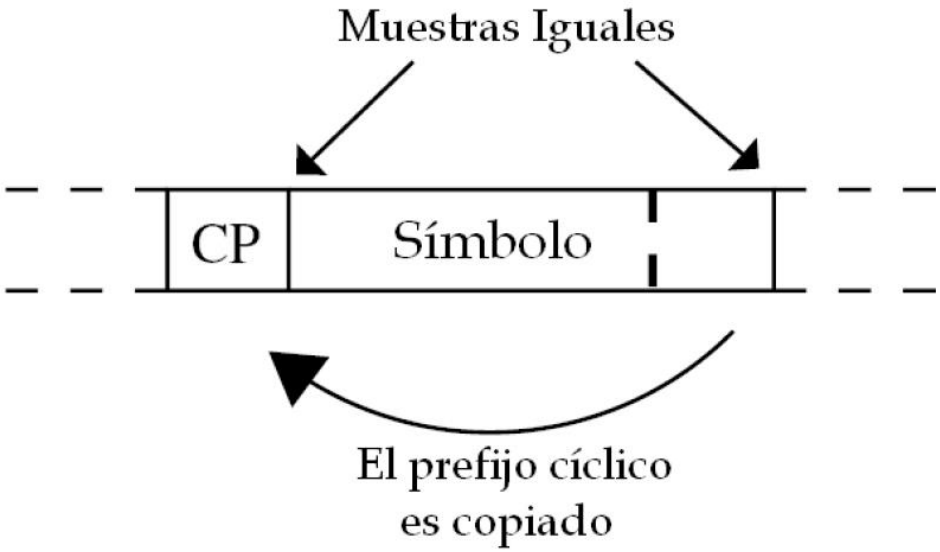

Figura 4.8 *Adición del prefijo cíclico*

Conversor Paralelo Serie

La matriz que entra a este bloque se transforma en un vector lineal antes de abandonar el transmisor y llegar al canal. Se recorre la matriz por símbolos para realizar la conversión, es decir, todos los símbolos de la matriz con su prefijo cíclico incluido se transmiten de forma contigua.

4.3.2. CANAL

Se trata del elemento a través del cual se envía la señal que sale del transmisor y cuyas características determinarán las variaciones que sufre la información. En el apartado 5.3 de MIMO se definirá como es este medio en un entorno de LTE con múltiples antenas.

4.3.3. RECEPTOR

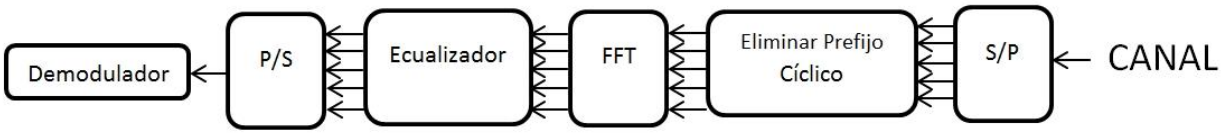

Figura 4.9 *Esquema de bloques del receptor*

Conversor Serie Paralelo

El vector lineal que sale del canal se transforma en una matriz con la misma estructura que la trabajada en el transmisor.

Eliminar Prefijo Cíclico

La extensión cíclica (añadida en el transmisor) es quitada del inicio de todos los símbolos.

FFT

Símbolo por símbolo se pasa del dominio del tiempo al de la frecuencia.

$$X(\omega) = \int_{-\infty}^{\infty} x(t) e^{-j\omega t} dt = \mathcal{F}\{x(t)\}$$

Figura 4.10 *Fórmula de la Transformada de Fourier*

Siendo x(t) una función continua en el dominio del tiempo y X(ω) la señal resultante en el dominio de la frecuencia.

Ecualizador

Los símbolos llegan al receptor con un cambio de amplitud y de fase debido al impacto del canal para las diferentes sub-portadoras. El ecualizador se encarga de obtener una matriz H que sirva para solventar los problemas surgidos mediante una estimación del canal basada en los símbolos de referencia introducidos en el transmisor.

Para la consecución de la matriz H debe realizarse una estimación de canal en frecuencia y tiempo, lo que supone una gran carga computacional y mayor complejidad para el sistema. Existen algoritmos que calculen por separado la estimación en la dirección frecuencial de la dirección temporal. Esta estimación será menos precisa si el ruido que hay en el canal es alto. En las situaciones en las que se dispone de un canal multitrayecto sin apenas ruido existente la estimación realizada es tan eficiente que permite prácticamente la corrección total de las variaciones surgidas.

En primer lugar mediante un algoritmo en el domino transformado de Fourier se realiza la estimación en el domino de la frecuencia. Hay que usar este método para todos los instantes temporales en los que existan señales piloto, es decir, para aquellas situaciones en las que los símbolos OFDM tengan sub-portadoras que sean señales de referencia.

El conjunto de los valores de las señales de referencia por instante temporal se denomina vector p, su tamaño es el número de pilotos existente en un símbolo OFDM.

$p = [\widehat{H}(0)\ \widehat{H}(N_f)\ \widehat{H}(2\ N_f)\ldots]$

El algoritmo consta de los siguientes pasos: [11]

- Se realiza la IDFT de todos los valores de las señales pilotos del símbolo.
  $\hat{h}s$ = IDFT(p)

- Se rellena la IDFT obtenida anteriormente con los ceros necesarios para que haya tantas muestras como sub-portadoras tenga el símbolo OFDM.
  $\hat{h}$ = [$\hat{h}$s 0 … 0]

- Se efectúa una DFT de $\hat{h}$. El resultado se corresponderá con la columna de la matriz H para el instante temporal del símbolo que se estaba usando y que contenía señales de referencia.
  $\widehat{H}$ = DFT($\hat{h}$)

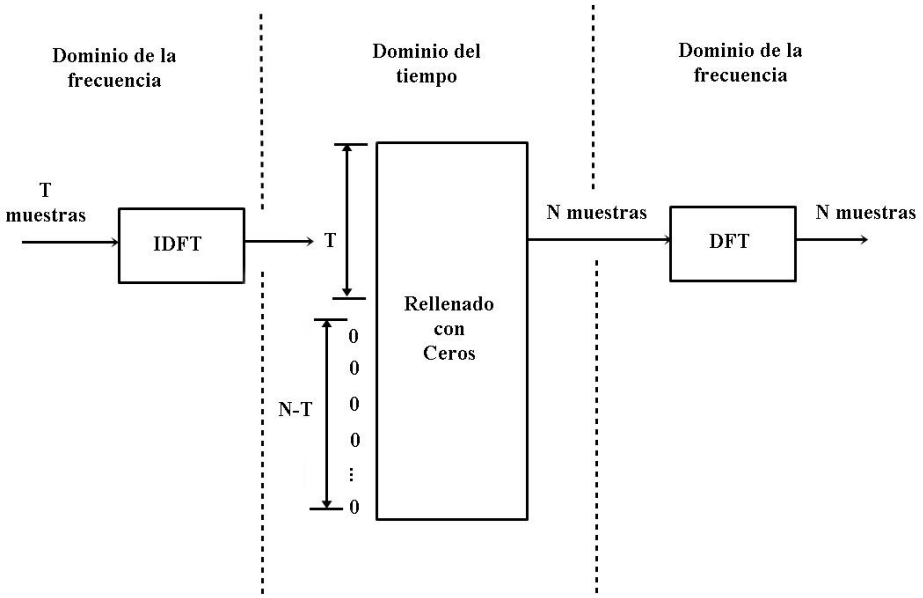

Figura 4.11 *Algoritmo para la estimación en el dominio transformado de Fourier*

Después de realizar la estimación en el domino de la frecuencia para cada una de las sub-portadoras que contenían señales pilotos, debe realizarse la estimación temporal para calcular el resto de valores de la matriz H. Se realiza una interpolación lineal por sub-portadoras, entre los valores estimados en el paso anterior de dos símbolos OFDM consecutivos.

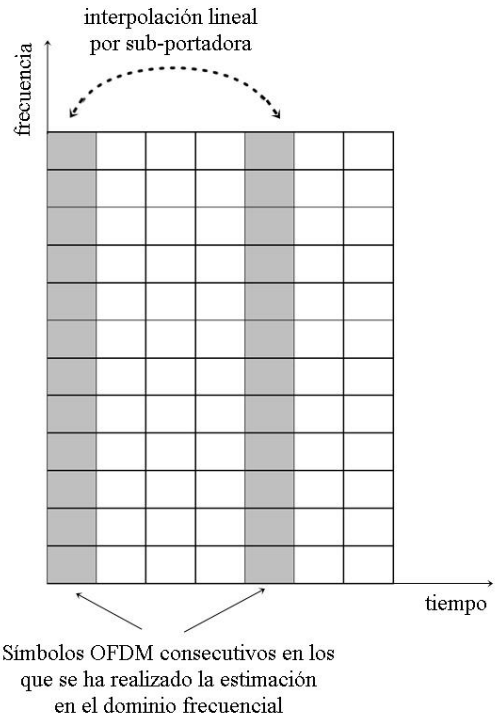

Figura 4.12 *Interpolación en el domino temporal*

La matriz H obtenida tiene el mismo tamaño que la matriz que ha llegado al ecualizador. Cada uno de sus elementos se corresponde con un número complejo basado en la respuesta de la estimación obtenida que indica los cambios de amplitud y fase necesarios para deshacer las variaciones perjudiciales sufridas por la información transmitida debido al multitrayecto y al ruido existente en el canal.

Para corregir de forma definitiva estos problemas cada posición de la matriz H obtenida divide a su posición correspondiente en la matriz que había llegado al ecualizador.

Paralelo Serie

La matriz de símbolos se convierte en un vector lineal siguiendo el proceso inverso realizado en el conversor S/P del transmisor y quitando las posiciones de las señales piloto, además de las frecuencias no usadas.

Demodulador

Los símbolos llegan al demodulador donde mediante las regiones de decisión correspondientes se determina cuáles son sus bits correspondientes.

4.4. TRAMA RADIO Y REJILLA DE RECURSOS

Como se ha explicado anteriormente en el conversor Serie – Paralelo del Transmisor, se envían unas determinadas señales pilotos en unas determinadas posiciones de la rejilla de recursos. Estas señales piloto o Reference Signal (RS) son unas señales predefinidas que permitirán calcular los cambios de amplitud y fase causados por el canal multitrayecto.

Según las características del sistema y sus requisitos serán utilizados diferente números de RB y de subportadoras. El siguiente cuadro obtenido del estándar indica alguna de las característas necesarias para diferentes anchos de banda.

ANCHO DE BANDA	1.25 MHZ	2.5 MHZ	5 MHZ	10 MHZ	15 MHZ	20 MHZ
Frecuencia de muestreo	192 kHz	3.84 MHz	7.68 MHz	15.36 MHz	23.04 MHz	30.72 MHz
	1/2 x 3.84	1 x 3.84	2 x 3.84	4 x 3.84	6 x 3.84	8 x 3.84
Tamaño FFT	128	256	512	1024	1536	2048
Número de PBRs	6	15	25	50	75	100
Número de subportadoras útiles	72	180	300	600	900	1200

La rejilla de recursos es la forma utilizada para organizar los recursos en OFDM para los distintos usuarios, con el objetivo de aprovechar de forma eficiente los recursos radio.

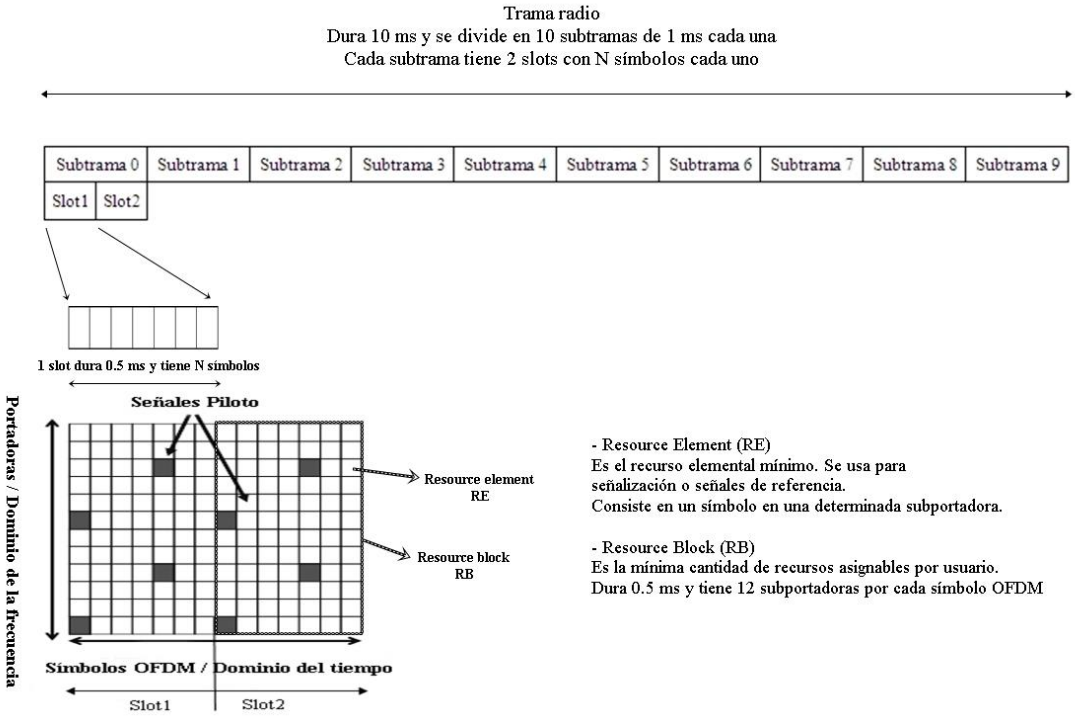

Figura 4.13 *Trama radio y rejilla de recursos de OFDM*

4.5. VENTAJAS E INCONVENIENTES

Ventajas OFDM:

Es un sistema robusto frente al multitrayecto y poco sensible a los desvanecimientos rápidos.

El receptor de banda base necesita poca complejidad.

La superposición de portadoras ortogonales entre sí optimiza la eficiencia espectral.

Existe una asignación flexible de ancho de banda. Esta flexibilidad implica repartir las frecuencias según la velocidad del servicio que se pretende ofrecer, para transmitir con unas tasas u otras.

Buenas propiedades espectrales y buen manejo de los múltiples anchos de banda.

Existe compatibilidad con las tecnologías avanzadas de antenas y receptores.

Inconvenientes OFDM:

Existe una alta relación entre la potencia media respecto a la potencia de pico, lo que implica que la batería se agote de manera más rápida, lo que resulta un impedimento para su uso en el enlace ascendente de LTE.

Se requieren bandas de guarda.

El emisor y el receptor deben estar sincronizados para saber cuál es la asignación de frecuencias que se está realizando en cada momento.

Esta tecnología es sensible a errores de frecuencia y sincronización.

5. MIMO

5.1. INTRODUCCIÓN Y ORIGENES

Esta tecnología consiste en la utilización de varias antenas para transmitir y/o recibir. Esto implica que como mínimo dos flujos de datos se transmiten y se reciben, ya sea para un único usuario o para varios usuarios.

MIMO surgió para perfeccionar parámetros mejorables en las comunicaciones móviles actuales tales como el alcance y el rendimiento e incrementar la velocidad de transmisión, este aumento de la capacidad es importante para disponer de una mayor eficiencia espectral en los sistemas de comunicaciones.

La utilización de múltiples antenas realmente empezó a estudiarse a comienzos del siglo XXI, aunque ya habían existido diversas aplicaciones para uso militar con anterioridad.

En la Release 7 de 3GPP se introdujo como nueva funcionalidad las antenas MIMO para HSPA+ (HSPA Evolucionado), constituyendo una importante novedad en el mundo de las comunicaciones móviles.

En la Release 8, donde fue definida la primera versión de LTE, se determina la implementación del sistema con varias antenas para un único usuario. En este caso el uso de múltiples antenas es un aspecto clave de la tecnología junto a OFDM desde el primer momento. En cambió con HSPA+, fue una mejora que se añadió posteriormente.

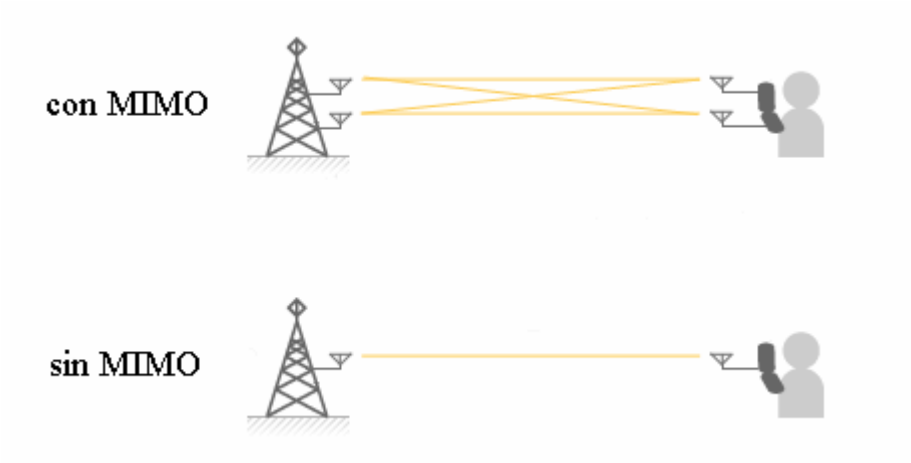

Figura 5.1: *Comparación entre el uso y no uso de la tecnología MIMO*

5.2. CONCEPTOS FUNDAMENTALES

El uso de una o varias antenas tanto en el transmisor o el receptor determina el nombre de las diferentes tecnologías:

- SISO: Una antena en cada extremo del radio enlace.
- SIMO: Una única antena en el transmisor y varias en el receptor.
- MISO: Múltiples antenas transmitiendo y una sola antena en el receptor.
- MIMO: Múltiples antenas tanto en el transmisor como en el receptor.

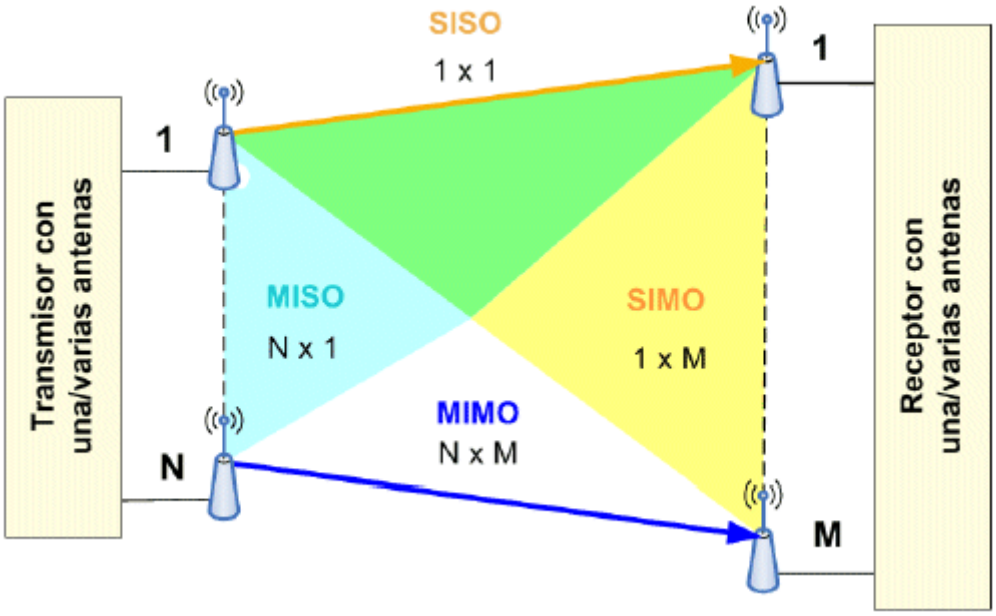

Figura 5.2: *Combinación de antenas en el transmisor y receptor*

El uso de MIMO mejora el rendimiento ya que se produce un aumento de la capacidad y la velocidad de transmisión, se cancelan interferencias y el sistema se vuelve más robusto.

Al usar un número determinado de antenas se multiplica la capacidad existente por ese número de antenas sin incrementar la potencia ni el ancho de banda, también se puede optar por transmitir la misma capacidad pero usando mucha menos potencia.

Ahora existirán múltiples canales entre las diferentes antenas del transmisor y el receptor. Hay que evaluar y representar el canal de forma matemática tratando el canal como si fuese una matriz, para poder recuperar las distintas señales transmitidas. Como se ha explicado con anterioridad esta función la realizará el ecualizador, ahora con mayor complejidad debido a la existencia de diferentes canales. [12]

MIMO consigue reducir los efectos perjudiciales surgidos a partir de los diferentes canales multitrayecto. La mayor robustez derivada del uso de múltiples antenas implica que el sistema se ve menos afectado a medida que aumenta el nivel de ruido existente en el canal, es decir, el aumento de los errores que surgen en la transmisión a medida que disminuye la relación señal a ruido es menor.

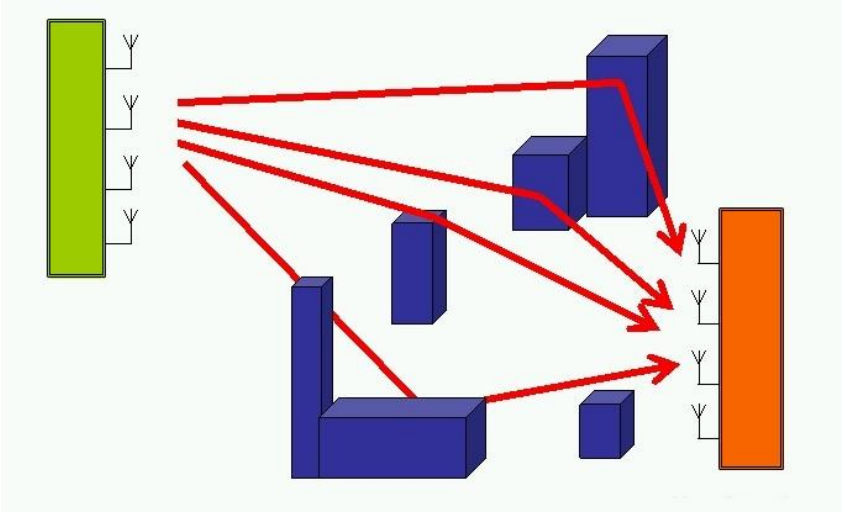

Figura 5.3: *Multitrayecto generado en los canales MIMO*

En la Release 8 mencionada con anterioridad, se especificaron que podrían usarse hasta 4 antenas en el enlace descendente. Antes en el apartado 4.4 se ha hablado acerca de la rejilla de recursos de OFDM. Ahora habrá una rejilla de recursos para cada antena con una ubicación diferente de señales piloto o RS para cada una de ellas. Donde en una antena exista una posición piloto en el resto de antenas esa posición en su rejilla de recursos no podrá usarse, ya que será utilizada en exclusiva para medir los efectos del canal sobre una determinada comunicación entre antenas.

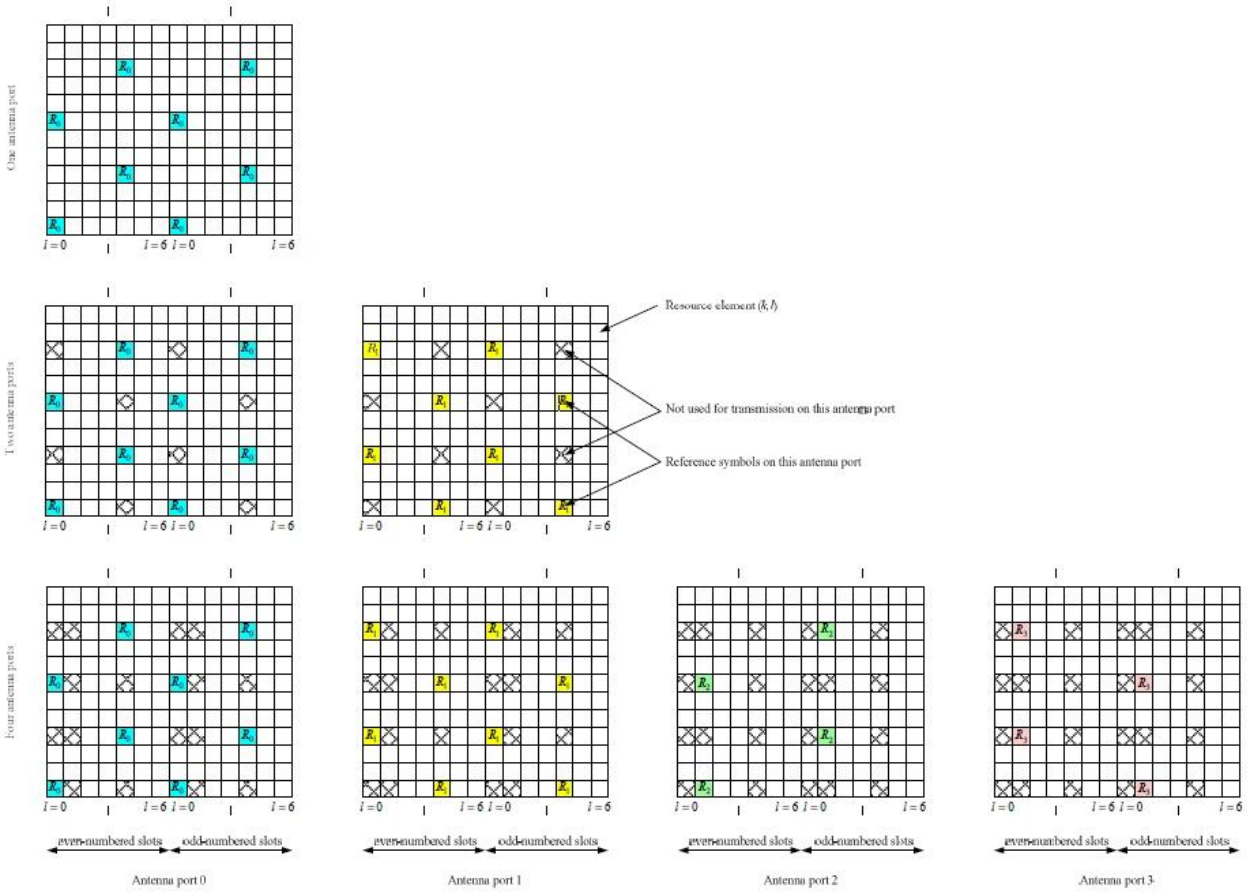

Figura 5.4: *Ubicación de símbolos piloto para diferentes antenas MIMO [10]*

5.3. RECEPTOR ZERO-FORCING

El uso de un receptor Zero-Forcing en los sistemas con múltiples antenas es necesario para recuperar las señales transmitidas y posteriormente en el demodulador obtener los símbolos que fueron enviados.

Es imprescindible usar este método ya que las señales OFDM que llegan al ecualizador contienen información superpuesta de todas las señales que se envían a través del transmisor. Esta mezcla de información es producida en el canal, donde los diferentes caminos existentes entre las antenas transmisoras y receptoras, llevan las señales OFDM que al llegar al receptor se combinan entre sí.

Las señales **y** que llegan al ecualizador son de la forma:

$$\mathbf{y = H*x + n}$$

Siendo **x** las señales OFDM transmitidas, **n** el ruido del canal y **H** la matriz con la estimación de los diferentes canales.

Se realiza la estimación de canal para poder obtener la matriz H, mediante el uso de los diferentes tipos de ecualizador con sus señales de referencia correspondientes. Con el cálculo de esta matriz, la obtención aproximada de las señales x enviadas desde el transmisor será más sencilla. La forma de conseguirlo es mediante la siguiente expresión:

$$\tilde{x} = \mathbf{H^{-1}*y}$$

Para ello habrá que realizar la pseudoinversa de la matriz H y multiplicarla por las distintas señales OFDM que llegan al ecualizador. De este modo se conseguirán las señales que fueron enviadas a partir del modulador del transmisor. [13]

A medida que el ruido n existente en el canal se eleva la detección de las señales x guardará menos similitudes con las transmitidas originalmente.

5.4. CANAL SCM

Las comunicaciones móviles requieren que para el diseño y la simulación de los sistemas de comunicaciones inalámbricos sea absolutamente necesario especificar un modelo de canal de propagación.

El canal que se va a desarrollar en este proyecto es un canal SCM (Spatial Channel Model). El código para la elaboración de este tipo de canal en Matlab ha sido realizado por 3GPP y por tanto no se considera como parte del sistema implementado ya que no ha sido desarrollado como parte del código elaborado para el presente proyecto.

El canal SCM consiste en un modelo geométrico o modelo de rayos que permite evaluar conceptos MIMO en sistemas de múltiples antenas. Fue desarrollado por el 3GPP con el objetivo de simular el sistema LTE antes de su implementación y poder conocer los requisitos para realizar una correcta planificación de la red. [14]

Para el uso de este canal hay que seleccionar el entorno en el que se quiera trabajar y determinar los parámetros característicos del mismo.

Existen unos parámetros de radio enlace, unos parámetros de configuración del modelo y los parámetros de la antena, como entradas que sirven para obtener las matrices del canal MIMO como salida.

La parte del radio enlace se encarga de caracterizar el enlace existente entre una estación base (BS) y una estación móvil (MS) para un enlace descendente y entre una BS y una MS en el caso del enlace ascendente. Es importante saber cuál de los dos enlaces se quiere implementar, ya que para realizar la simulación la configuración del canal puede ser tanto ascendente como descendente.

La configuración del modelo tendría en cuenta la comunicación entre varias estaciones móviles y varias estaciones base, además de las interferencias que puedan surgir entre ellas. En este trabajo se realizará la simulación del enlace descendente, es decir, las antenas transmisoras serán las estaciones base y establecerán una comunicación con las estaciones móviles que se corresponden con las antenas receptoras.

La salida consiste en un array multidimensional que contiene las respuestas al impulso del canal para un número predefinido de radio enlaces entre el transmisor y el receptor.

Estos últimos párrafos consisten en una explicación breve de los parámetros a indicar para generar un canal válido para LTE.

Todos los parámetros existentes, ya sean para la configuración del enlace, del modelo o de las antenas, pueden ser establecidos como se desee. Algunas de estas configuraciones de entrada, como la orientación de las estaciones o la distancia entre la estación base y la estación móvil entre otros, pueden ser modificados si así se desea. Pero en algunas situaciones, como en este proyecto fin de carrera, no es de mucha utilidad preocuparse por indicar este tipo de configuraciones y es más sencillo indicar estos parámetros de forma aleatoria ya que la utilidad de generar un canal SCM es usarlo para establecer una comunicación de enlace de bajada en LTE y no realizar un estudio de las características de este canal.

Por lo tanto sólo será necesaria la configuración de los parámetros que definen los tipos de sistemas MIMO (SISO, MIMO 2x2 o MIMO 4x4) para obtener la matriz de salida a usar en el sistema desarrollado. Para ello la configuración del canal será suficiente con seleccionar el número de antenas en cada extremo del radio enlace y el número de multitrayectos producidos.

6. DESCRIPCIÓN DEL SISTEMA DESARROLLADO

6.1. HERRAMIENTAS UTILIZADAS

Este proyecto, como ya se ha mencionado, tiene el objetivo de simular un sistema MIMO-OFDM en el enlace descendente de LTE usando MATLAB®.

MATLAB® (MATrix LABoratory) es una herramienta software matemática con un lenguaje de alto nivel y un entorno interactivo que posee un lenguaje de programación propio. Con este programa es factible el desarrollo de algoritmos, el análisis de datos y la creación de aplicaciones.

Dentro de las operaciones matemáticas destaca el uso de las matrices, algo fundamental en la realización de este proyecto.

Este software destaca por su gran uso como herramienta de cálculo técnico, es usada por científicos de la educación y de la industria e ingenieros en todo el mundo. Se estima que en 2004 era usado por más de un millón de personas.

Destaca su uso en múltiples aplicaciones como sistemas de control, pruebas y medidas, procesado de imagen y video, finanzas computacionales, biología computacional… o como en el presente trabajo fin de grado para procesamiento de señales y comunicaciones. [15]

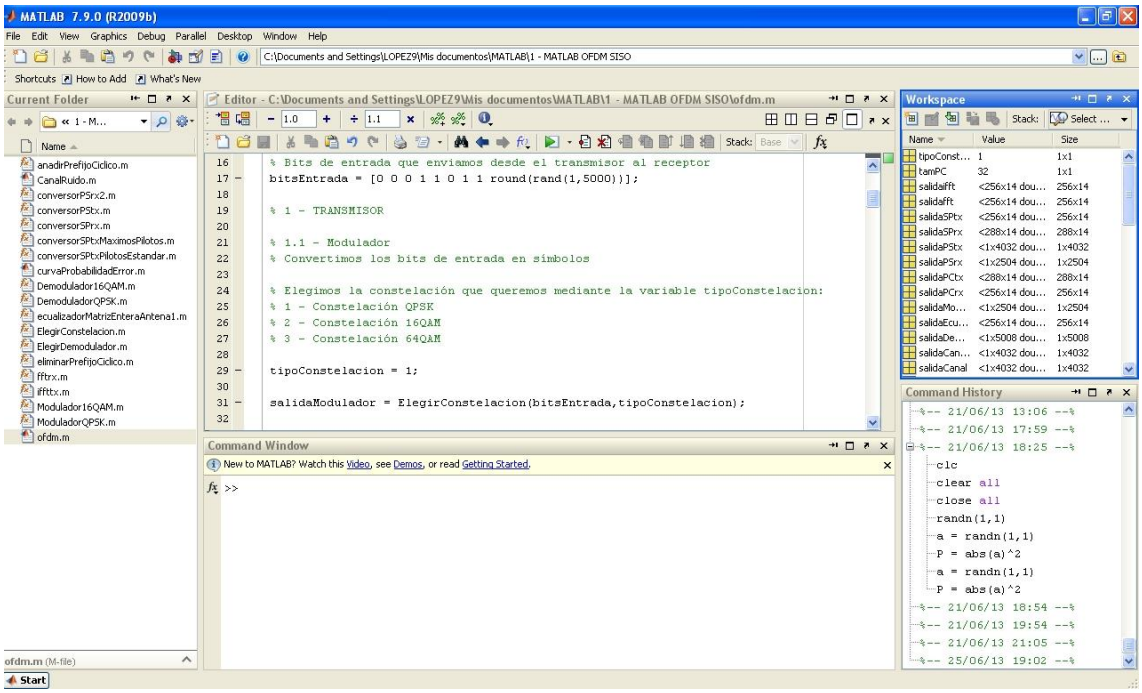

Figura 6.1 *Entorno de MATLAB®*

6.2. SISTEMA DESARROLLADO

Este proyecto se resume en tres partes diferenciadas, en la primera parte se realiza la simulación de un sistema SISO con una antena transmisora y una antena receptora, la segunda parte consta de dos antenas transmisoras y dos receptoras para desarrollar un sistema MIMO 2x2 y en la tercera parte se usan cuatro antenas en la transmisión y otras cuatro en la recepción para simular un MIMO 4x4.

Las características desarrolladas en este proyecto están basadas en la Release 8 de 3GPP, todos los aspectos implementados tienen como Marco Regulador Técnico este primer estándar de LTE y sus limitaciones para el enlace descendente. En este proyecto se usa hasta el máximo número de antenas posible en cada extremo del radioenlace, en este caso cuatro. Todas las partes disponen de la opción de elegir entre el uso de una modulación QPSK, 16-QAM ó 64-QAM. También hay que especificar cuál es el tamaño de la FFT a usar durante toda la simulación, siendo sus valores posibles 128, 256, 512, 1024, 1536 y 2048.

La comunicación siempre comienza con la selección del número de bits aleatorios a transmitir y determinando cual será el tipo de modulación y el tamaño de FFT elegidos. Estas características de la retransmisión serán indicadas en el script de Matlab que se desee ejecutar (existe un script para el sistema SISO, otro para MIMO 2x2 y uno para la tecnología MIMO 4x4). Cada script invoca un número determinado de funciones.

A medida que aumenta el tamaño de la cadena de bits a enviar se incrementa el tiempo necesario para ejecutar la simulación.

A continuación se describirá cómo se ha implementado cada uno de los sistemas anteriores y como se ha procedido a aplicar la teoría de las tecnologías OFDM y MIMO explicadas en los apartados previos para realizar la programación en Matlab y conseguir simular el enlace de bajada de LTE de forma correcta.

La descripción de las funciones realizadas en el apartado 6.2.1 con la tecnología SISO consistirán en la base de los sistemas implementados con más de una antena, por lo tanto en la siguiente parte referente al uso de una única antena transmisora y receptora se encontrará la explicación fundamental para todos los sistemas desarrollados en este proyecto.

En el resto de apartados de esta sección referentes al uso de múltiples antenas se añadirán los cambios introducidos respecto al uso de una única antena en cada punto de la transmisión.

En cuanto al canal que se usará en la comunicación será el SCM que ya fue explicado en el apartado anterior, caracterizado para cada uno de los sistemas en función del número de antenas a incluir en la tecnología y determinando el número de multitrayectos que se pretende disponer en la simulación. No sé realiza una explicación del mismo en el sistema desarrollado ya que se ha utilizado un código elaborado por 3GPP.

El uso de este canal es importante de cara a la obtención de las gráficas de los resultados de simulación y para establecer las conclusiones, ya que la matriz obtenida a partir de las funciones que desarrollan este canal se adecuan de forma muy precisa al canal existente en la realidad.

6.2.1. SISO

Los bits de entrada se convierten en una señal modulada cuando llegan a la función *ElegirModulador* que selecciona el modulador usado mediante la variable *tipoConstelacion*. Esta variable es la que se modifica en el script para indicar cuál es la modulación usada, si la variable es 1 se invocará la función *ModuladorQPSK*, si es igual a 2 se ejecutará *Modulador16QAM* y en caso de que la variable tenga un valor de 3 la función elegida será *Modulador64QAM*.

Se define el tamaño de FFT usado mediante la variable *Nfft*.

Cuando se llega al *conversorSPtxPilotosEstandar*, el objetivo es transformar el vector lineal de la señal modulada en una matriz tal y como se explicó en la teoría de OFDM en el apartado 4.

Mediante la variable *P* de valor 1 se indica como es la señal de referencia a introducir en la matriz. Es necesario insertar las señales de referencia en la matriz, poniendo mucho cuidado en que la inserción se produzca en las posiciones correctas definidas por el estándar (figura 6.2)

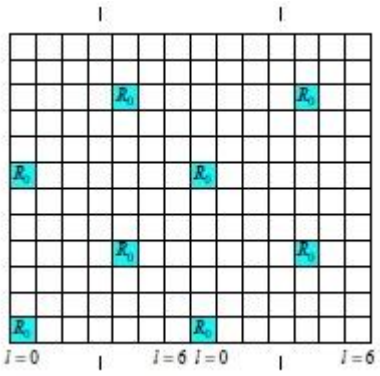

Figura 6.2 *Posiciones piloto SISO*

La matriz obtenida constituye la rejilla de recursos comentada en el aspecto teórico. El número de frecuencias (en este caso las filas) vendrá determinado por el tamaño de la FFT elegido, y el régimen temporal vendrá determinado por el número de columnas necesarias para realizar la conversión y poder incluir toda la información del vector lineal en la matriz a transmitir además de las señales piloto necesarias. Cuando se haya introducido toda la información el resto de espacios libres en la última columna serán completados con 0´s y se seguirá manteniendo la introducción de las señales de referencia en sus posiciones correspondientes.

No todas las frecuencias de la rejilla de recursos son utilizadas, existen algunas frecuencias de guarda en la parte superior e inferior de la matriz. El número de

frecuencias de guarda depende del tamaño de FFT utilizado y está definido en el estándar. Este valor es almacenado en la variable *GP* que será utilizada de forma recurrente.

La manera de desarrollar esta función en el código consiste en calcular en primer lugar cual es el tamaño de la matriz. Se determina el número de columnas a partir del tamaño de FFT (que constituye el número de filas) y a partir de la cantidad total de símbolos a incluir en la matriz. Estos símbolos consistirán en toda la información del vector que ha llegado al conversor Serie – Paralelo además de los símbolos de referencia totales que hay que añadir. Es muy importante conocer las dimensiones de la matriz, ya que si la matriz resultante fuese más pequeña de lo necesario parte de la información a transmitir no podría incluirse en la matriz y por tanto se perdería.

A partir del conocimiento del tamaño correcto de la matriz, se pondrán 0´s en las posiciones de las frecuencias no útiles y se incluirá el valor de la variable *P* en las posiciones donde sea conveniente enviar las señales de referencia según se define en el estándar. La forma de realizar esta operación es desplazarse por la matriz (en la zona de las frecuencias útiles) usando como unidad mínima la rejilla de la figura 6.2. Por último se recorre la información del vector rellenando el resto de elementos de la matriz columna por columna teniendo mucho cuidado de incluir estos símbolos en la parte de las frecuencias útiles que no son señales piloto. Por último, si es necesario, se rellanan con 0´s los espacios no completados de la última columna de la matriz obtenida.

Durante la elaboración de este trabajo se valoró la posibilidad de realizar la simulación usando el máximo número de frecuencias útiles posibles en lugar de las definidas por el estándar. Las razones para el planteamiento de esta solución alternativa se basaron en que al existir un número mayor de frecuencias útiles existirían más señales piloto por símbolo para la estimación del canal. Por lo tanto, al disponer de más señales piloto la estimación es mejor y los errores serían menores.

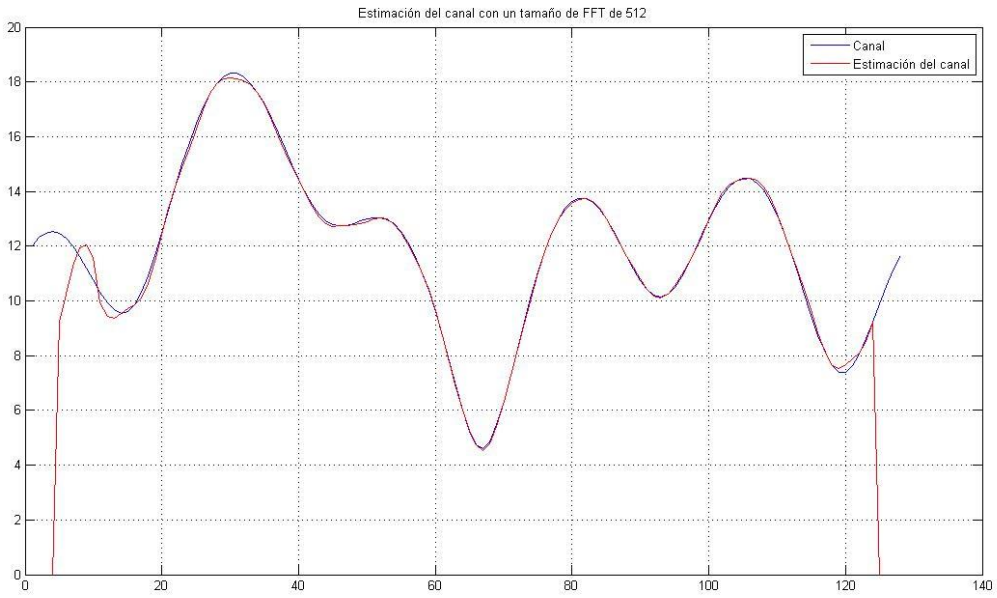

Figura 6.3 *Estimación del canal con el máximo número de frecuencias útiles*

Este número de frecuencias máximas viene determinado por el mayor múltiplo de 12 (12 es el número de sub-portadoras del RB de la rejilla de recursos) que cabe en una fila de la matriz definida.

Finalmente al realizar diversas retransmisiones se pudo comprobar que en una comunicación sin ruido, tal y como se había pensado en un principio, la estimación del canal era mucho mejor (Figura 6.3) y por tanto los fallos producidos eran ligeramente menores que usando las frecuencias útiles determinadas por la Release 8. Sin embargo con la posterior introducción de ruido en el canal se constató que los errores a medida que disminuía la relación señal a ruido aumentaban en mayor cantidad que usando las frecuencias útiles del estándar lo que convertía a los diferentes sistemas en poco robustos frente al ruido. Por lo tanto está opción de aumentar las frecuencias útiles al máximo posible para los diferentes tamaños de FFT fue descartada después de varias pruebas.

El siguiente paso es realizar una IFFT de todos los componentes de la matriz recién obtenida, mediante *ifft*tx se pasa del dominio de la frecuencia al temporal. Se realiza símbolo por símbolo, es decir, se realiza una IFFT en cada una de las columnas.

El proceso de añadir el prefijo cíclico comienza por la definición del valor del prefijo cíclico, mediante *tamPC = Nfft/8*. Se ha determinado que este valor sea un octavo del valor del tamaño de FFT usado para que de forma segura no se produzca ISI, ya que este valor es mayor que la respuesta impulsiva del canal. El prefijo cíclico como se ha insistido en la parte teórica es fundamental en OFDM e implica que se añadan más frecuencias (filas) a la matriz.

La adición se produce mediante un bucle que recorre cada una de las columnas de la matriz e invoca a la función *anadirPrefijoCiclico* encargada de copiar el final de la columna en la que se encuentra en el principio de la misma. El tamaño de la parte copiada es la variable *tamPC*.

El último proceso del transmisor consiste en ejecutar la función *conversorPStx* para transformar la matriz en un vector lineal y poder realizar la transmisión por el canal. El cambio a vector lineal se realiza introduciendo en el nuevo vector todas las posiciones de la matriz columna por columna, es decir, los símbolos existentes en una misma columna se introducen de forma consecutiva en el vector. Por lo tanto, en el vector obtenido cada uno de los símbolos (incluido su prefijo cíclico) estarán situados de forma contigua sin que haya una separación de la información contenida en ellos.

La forma de implementarlo consiste en recorrer la matriz columna por columna y desplazar la información de cada una de las columnas en el vector que se pretende enviar hacia el canal. Los símbolos de la matriz son copiados en cada columna de arriba hacia abajo y se introducen de izquierda a derecha en el vector.

La información llega al medio en el que realiza, determinada por la matriz obtenida mediante las funciones del canal SCM.

Antes de añadir ruido, es necesario normalizar el canal, para ello se debe comparar la potencia de la señal obtenida en el modulador y la que sale del canal sin normalizar. La diferencia entre ambas será el valor que multiplique la matriz del canal a

usar (en este caso vector al existir una única antena). De este modo cualquier canal que se quiera utilizar podrá ser normalizado y la potencia de la señal después de pasar por el canal será la misma que al salir del modulador al comienzo del transmisor.

Después de la normalización del canal se procede a incorporar el ruido, para ello se genera aleatoriamente un ruido gaussiano *ni* con una amplitud *ni1* y que toma el valor que se considere apropiado para realizar las diferentes mediciones. A medida que el tamaño de esta amplitud sea mayor la potencia del ruido introducido en el canal también lo será.

Antes de que la información deje el canal, se mide la potencia de la señal que sale del canal normalizado sin ruido añadido (es la misma potencia que la señal que deja el modulador) y también se determina la potencia del ruido añadido, con estas dos cantidades se calcula el valor de la relación señal a ruido (SNR) siendo el cociente entre la potencia de la señal y la potencia del ruido. Este valor se encuentra en unidades naturales y será indispensable convertirlo a unidades logarítmicas calculando el logaritmo y multiplicando por 10. Estos cálculos del SNR serán fundamentales para la realización de las curvas de probabilidad de error cuya elaboración será explicada en este mismo apartado dentro del punto 6.2.4.

Después de que la señal viaje a través del canal, la información llega al receptor, allí la primera función a usar será el *conversorSPrx* convierte el vector lineal obtenido a través del canal en una matriz con la misma estructura que la desarrollada en el transmisor con el prefijo cíclico incluido.

Las dimensiones de esta matriz vienen definidas por un número de filas que consiste en el tamaño de la FFT más el tamaño del prefijo cíclico. Se recorre el vector de izquierda a derecha copiando un número de símbolos concreto definido por el tamaño de las filas de la matriz, estos símbolos se incluyen de arriba hacia abajo en la columna de la matriz que corresponda (a medida que se recorre el vector también se recorre la matriz columna por columna).

Posteriormente se procede a la eliminación del prefijo cíclico. Como resultado la matriz pasa a tener el número de sub-portadoras (filas) definidas por el tamaño de la FFT, es decir, el mismo tamaño que la rejilla de recursos obtenida tras el conversor Serie – Paralelo del transmisor. Este procedimiento es realizado por la función *eliminarPrefijoCiclico* que actúa en un bucle recorriendo cada una de las columnas de la matriz y quitando de la parte inicial de cada columna la cantidad determinada por la variable *tamPC*.

El siguiente proceso, es uno de los más complejos de desarrollar y consiste en la implementación del ecualizador a través de *ecualizadorMatrizEnteraAntena1*. La misión de esta función de Matlab es obtener una matriz H para corregir los cambios de amplitud y fase originados por el canal. Para poder obtener esta función es necesario seguir los pasos explicados en la teoría de OFDM en el punto 4.3.3, realizando en un primer momento una estimación en el dominio transformado de Fourier y más tarde una estimación en el dominio temporal.

Tras obtener la matriz H, cada una de sus posiciones divide a su posición correspondiente de la matriz que ha llegado al ecualizador. De esta forma se corrigen los cambios de amplitud y fase sufridos por la señal durante el canal.

La matriz que sale del ecualizador, ya con la corrección de los cambios realizada gracias a la estimación del canal, llega a la función *conversorPSrx2* donde la información vuelve a expresarse de forma lineal.

Para su elaboración es necesario eliminar en un primer momento las frecuencias no usadas de la matriz. Después se recorre la matriz de arriba abajo columna por columna se insertan los símbolos de información en el nuevo vector de izquierda a derecha sin añadir las señales de referencia ya que estas no deben ser incluidas. Después de este proceso se eliminan del vector los 0´s que fueron añadidos en el Conversor Serie - Paralelo del transmisor para completar la matriz. De esta forma el tamaño del vector será igual que el vector originado después del modulador en el bloque del transmisor. Es muy importante eliminar las posiciones correctas de la matriz ya que de no ser así se perdería parte de la información a transmitir y los errores aumentarían, además de incluir la información en su posición correcta dentro del vector, para ello es necesario establecer de forma idónea la ubicación de todas las señales de referencia y saber el número de las frecuencias de guarda existentes, definido por la variable *GP*, y que es usado en gran parte de las funciones del script.

Esa señal pasa la función *ElegirDemodulador* que selecciona la demodulación a usar mediante la variable *tipoConstelacion* definida al comienzo del script. Dependiendo del valor de la variable será seleccionada la función *DemoduladorQPSK* cuando la variable toma valor 1, la *Demodulador16QAM* cuando el valor es 2 o la *Demodulador64QAM cuando es 3*, que serán encargadas de expresar la señal otra vez en forma de bits, lo que constituye la información recibida por el receptor.

Tanto las funciones de los moduladores como de los demoduladores usadas han sido realizadas de forma manual sin haber recurrido a las funciones preestablecidas de Matlab.

Por último estos bits son comparados con los bits de entrada del transmisor, para determinar los fallos y el porcentaje de errores de la retransmisión. Es importante relacionar los errores surgidos con la relación señal a ruido de la comunicación.

6.2.2. MIMO 2x2

Este sistema tiene su base fundamental en el sistema OFDM desarrollado en la tecnología SISO, explicada en el punto anterior, siendo algunas de sus funciones comunes en ambas tecnologías.

Los bits de entrada se dividen en dos partes que se reparten para cada una de las dos antenas existentes mediante dos flujos de información. Estos flujos de información se dirigen de forma simultánea al modulador.

Al igual que en SISO se selecciona el tipo de modulador mediante la variable *tipoConstelacion*. Este modulador será el usado para ambos flujos y como resultado se obtienen dos señales moduladas a partir de las dos cadenas de bits, usando las funciones *ModuladorQPSK*, *Modulador16QAM* o *Moduladora64QAM*.

La primera variación del sistema surge cuando se llega a los conversores Serie – Paralelo. En este caso al disponer de dos antenas transmisoras y dos antenas receptoras se dispondrá de dos tipos de rejilla de recursos diferentes con diferentes posiciones de las señales de referencia según lo explicado en la teoría MIMO del apartado 5 y como se puede apreciar en la figura 6.4. Por lo tanto existirán dos funciones diferentes *conversorSPtxAntena1PilotosEstandar* y *conversorSPtxAntena2PilotosEstandar*, para cada una de las antenas del sistema. Las señales piloto a introducir continúan siendo definidas mediante la variable *P* que equivale a 1. Las posiciones usadas para señales de referencia se corresponden con posiciones no usadas en la otra rejilla (se introducen 0´s en esas posiciones) y viceversa. Se mantiene el uso de las frecuencias útiles definidas por el estándar según el tamaño de FFT y también la adición de 0´s hasta completar las matrices.

Básicamente, las diferencias consisten en que en esta situación hay que tener en cuenta también el número de posiciones no usadas para calcular el tamaño de la matriz, el desplazamiento realizado en cada una de las matrices se basa en las unidades mínimas de las figuras 6.4, se insertan 0´s en las posiciones no usadas y para completar la matriz con la información del vector es necesario percatarse de que aparte de en las posiciones de las señales piloto tampoco hay que insertar los símbolos en las posiciones no usadas ni en las frecuencias no usadas. La forma de rellenar la matriz también se realiza recorriéndola con un bucle columnas por columna y de arriba hacia abajo.

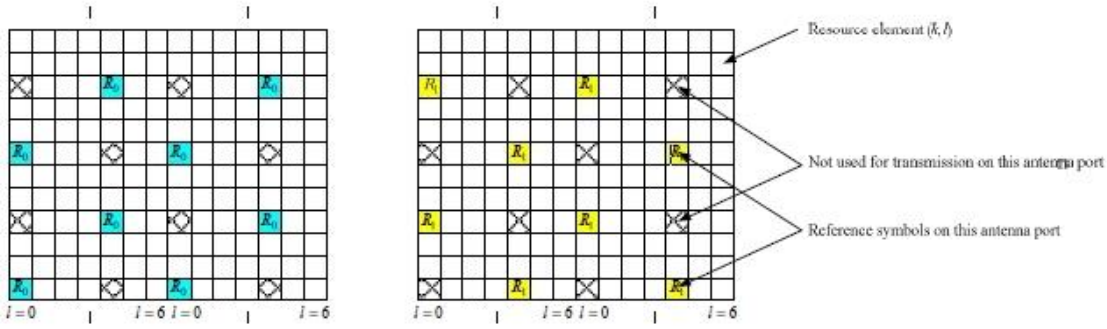

Figura 6.4 *Posiciones piloto MIMO 2x2*

Posteriormente se realiza la IFFT, la adición del prefijo cíclico y la conversión Paralelo – Serie para los dos flujos de información siguiendo los mismos procesos que los usados en la tecnología SISO. Estos son los últimos pasos efectuados en el transmisor.

Al finalizar las funciones del transmisor, se incluyen los vectores de los diferentes canales SCM elaborado mediante las funciones Matlab de 3GPP. Existen cuatro vectores diferentes debido a las 2 antenas en el transmisor y las otras 2 en el

receptor y a las posibles comunicaciones entre las mismas. Esta serie de canales provoca que en cada uno de los flujos de información que salen del canal se mezcle parte de la información de una antena con la información de la otra.

Al igual que en el apartado anterior es necesario normalizar los canales mediante el control de las potencias de las señales para que la señal al atravesar el canal disponga de la misma potencia que antes de entrar al mismo.

Se añaden dos ruidos gaussianos aleatorios diferentes a las salidas del canal siguiendo el mismo proceso que en el canal del sistema con una única antena definido en el punto 6.2.1. El mismo valor de amplitud que se utiliza para ambos ruidos determinará el valor de la relación señal a ruido de la simulación realizada.

Ya en el receptor se utilizan funciones idénticas a las desarrolladas en SISO para efectuar la conversión Serie – Paralelo, la eliminación del prefijo cíclico y la FFT en cada uno de los flujos de información.

Debido a los dos tipos de rejillas existentes se dispone de dos ecualizadores diferentes, las funciones a usar son *ecualizadorMatrizEnteraMIMO2Antena1* y *ecualizadorMatrizEnteraMIMO2Antena2*.

Los ecualizadores realizan la estimación del canal basándose en las posiciones piloto de la rejilla de su antena correspondiente. Cada una de las funciones actúa en los dos flujos de información para estimar los cuatro canales existentes en la comunicación entre las cuatro antenas.

Se obtienen cuatro matrices diferentes, a partir de la combinación de todas ellas se realiza una pseudoinversa que permite corregir las variaciones del canal y aproximar la información de cada flujo que llega al ecualizador a la información enviada anteriormente a través del transmisor, aparte de solucionar la mezcla de información producida en el canal. Este método se basa en los conceptos teóricos del receptor Zero-Forcing visto en la teoría MIMO del apartado 5.3 y para realizarlo hay que multiplicar la pseudoinversa de la matriz de estimación del canal por las señales llegadas al ecualizador, de este modo, se obtienen las señales enviadas desde el transmisor.

La forma de realizar la conversión Paralelo – Serie en el receptor varia respecto al sistema SISO, aunque se usa una misma función, *conversorPSrxMIMOAntena1*, para cada una de las matrices de información destinadas a las antenas receptoras.

El único aspecto a tener en cuenta es que ahora se eliminan más posiciones de la rejilla de recursos, es decir, en el vector lineal no solo no se incluyen las posiciones de las señales piloto sino que tampoco se cuentan las posiciones no usadas; aparte de las frecuencias no usadas y de cada grupo de 0´s añadidos en la última columna de cada una de las matrices.

Posteriormente las dos señales se convierten en los bits de salida a partir del uso de la función del demodulador elegido al comienzo de la comunicación mediante la variable *tipoConstelacion*.

Finalmente son calculados los fallos de cada flujo de información de forma independiente en relación con los bits de entrada que fueron repartidos al inicio del

transmisor entre las dos antenas. A partir de los porcentajes de errores de cada una de las antenas se realiza un promedio de los mismos para calcular la probabilidad de fallos del sistema.

6.2.3. MIMO 4x4

Teniendo como base OFDM, este sistema es prácticamente idéntico a la tecnología MIMO de 2 antenas trabajando con cuatro flujos de información. En esta nueva implementación existen cuatro rejillas de recursos diferentes lo que implica la ampliación del número de conversores Serie-Paralelo en el transmisor y del número de ecualizadores en el receptor hasta cuatro.

Los bits totales de entrada que se desean enviar se reparten en cuatro flujos entre las cuatro antenas transmisoras. Estos grupos de bits llegan al bloque del modulador de forma conjunta.

Las cadenas de bits se modulan usando un mismo modulador que es seleccionado en función del valor de la variable *tipoConstelacion*. Esta variable invoca *ModuladorQPSK*, *Modulador16QAM* o *Modulador64QAM* para obtener las cuatro señales que viajan a través del transmisor.

Cada señal modulada llega a su respectivo conversor Serie – Paralelo, donde se produce la inserción de las señales piloto con valor *P = 1*, también se añaden las posiciones no usadas con valor nulo, además de las frecuencias de guarda en función del tamaño de FFT y los 0´s hasta completar la matriz. Estas posiciones se basan en la ubicación definida por el estándar según la imagen 6.5. Existen cuatro conversores diferentes: *conversorSPtxAnt1PilotosEstandar*, *conversorSPtxAnt2PilotosEstandar*, *conversorSPtxAnt3PilotosEstandar* y *conversorSPtxAnt1PilotosEstandar*. La existencia de estas cuatro funciones se debe a la nueva disposición de las rejillas de recursos donde se dispone de un mayor número de posiciones no usadas. El objetivo sigue siendo obtener la matriz a partir del vector lineal de información.

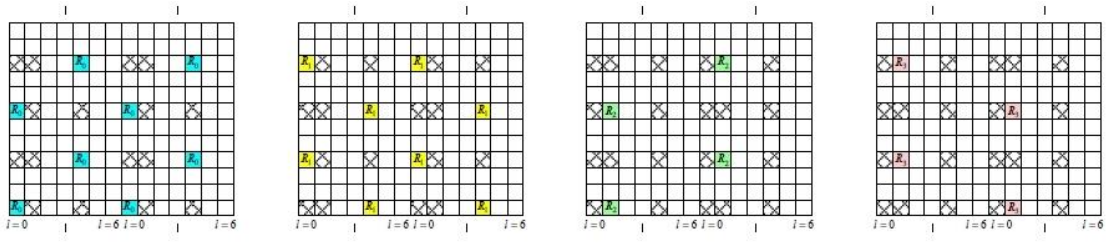

Figura 6.5 *Posiciones piloto MIMO 2x2*

Posteriormente se realiza la IFFT, la adición del prefijo cíclico y la conversión Paralelo – Serie a cada una de las matrices de información antes de la llegada al canal.

Tras obtener los vectores de cada uno de los canales a partir de las funciones de 3GPP y haber comprobado que la normalización de los canales es correcta mediante las mediciones de potencia de las diferentes señales, se añaden cuatro flujos de ruido gaussiano diferentes a la información que sale del canal SCM.

Los cuatro vectores de información que salen del canal contienen información mezclada de todos los flujos procedentes de las antenas transmisoras.

La información llega al receptor, donde se usarán las funciones *conversorSPrx*, *eliminarPrefijoCiclico* y *fftrxMIMO*, al igual que en los sistemas anteriores, se realiza la FFT para cada columna de las cuatro matrices.

Después se obtienen las cuatro matrices H que realizan la estimación del canal a partir de los cuatro ecualizadores: *ecualizadorMatrizEnteraMIMO2Antena1*, *ecualizadorMatrizEnteraMIMO2Antena2*, *ecualizadorMatrizEnteraMIMO2Antena3* y *ecualizadorMatrizEnteraMIMO2Antena4*.

Se obtienen dieciséis matrices diferentes, ya que cada ecualizador ha actuado en todas las matrices de información. A partir de las dieciséis matrices se aplica el Zero-Forcing, mediante la multiplicación de la pseudoinversa de todas las matrices H juntas por las señales recibidas en el ecualizador. De esta forma se consiguen las señales OFDM del transmisor con las variaciones producidas por el canal y el ruido mitigadas.

En el receptor, el conversor Paralelo – Serie es similar al usado en la tecnología MIMO 2x2 pero es necesario eliminar un mayor número de posiciones, debido a que el uso de más antenas conlleva disponer de más posiciones no usadas en la rejilla de recursos. La función encargada de realizar este cometido se llama *conversorPSrxMIMO4antenas* y su uso es común para todas las matrices que salieron de sus respectivos ecualizadores.

Posteriormente los cuatro vectores lineales que salen del conversor Paralelo – Serie del receptor son demodulados de forma independiente mediante la función *ElegirDemodulador* que selecciona el tipo de demodulador en función de la variable *tipoConstelación* definida a comienzos del script e invoca si el valor que toma la variable es 1 a *DemoduladorQPSK*, si es 2 a *Demodulador16QAM* y si fuese 3 a la función *Demodulador64QAM*. El resultado será la aparición de cuatro cadenas de bits que constituyen la información recibida.

Al igual que en los dos sistemas anteriores hay que comparar estos bits con los enviados a comienzos de la transmisión para comprobar los errores producidos y los porcentajes de los mismos en cada una de las antenas y posteriormente en todo el sistema.

6.2.4. Obtención de las gráficas

Tras la implementación de todas las tecnologías el siguiente paso es la elaboración de las gráficas de validación y simulación que serán mostradas y comentadas posteriormente en los capítulos 7 y 8.

Existen dos tipos de gráficas a desarrollar, aquellas que indican la estimación del canal que realiza el ecualizador y curvas de probabilidad de error.

Respecto a la elección del tamaño de la cadena de bits de entrada para la obtención de las gráficas es necesario saber que a medida que aumenta la longitud del vector de bits a transmitir se incrementa la precisión para determinar los errores existentes.

Para la elaboración de las curvas de probabilidad de error es necesario el uso de un elevado número de bits de entrada para conseguir que las gráficas obtenidas sean lo más exactas posibles y que las curvas obtenidas tengan una mayor suavidad. Esta mayor calidad de las curvas obtenidas tiene el inconveniente de que el tiempo de ejecución se eleva al tener que enviar una mayor cantidad de bits.

En cambio para las gráficas que reflejan la estimación del canal no es necesario disponer de un número de bits tan alto como para la realización de las curvas aunque tampoco puede usarse un número reducido ya que las prestaciones de la simulación descenderían.

El primer tipo de gráficas consiste en una comparación del canal en el dominio frecuencial con la estimación del canal de uno de los símbolos de la matriz H. Cada uno de esos símbolos consiste en una estimación del canal para su respectivo instante temporal, aunque con usar uno de los símbolos es suficiente ya que todos ellos guardan un aspecto muy semejante.

Para la elaboración de las curvas de probabilidad de error es necesario realizar diferentes retransmisiones y determinar la probabilidad de error existente en cada una de las comunicaciones establecidas.

En cada nueva retransmisión se añade más ruido con respecto a la anterior comunicación. En total para la obtención de cada una de las gráficas se realizan 49 retransmisiones con un rango de valores de relación señal a ruido (SNR) entre 5.25 dB y 24.7 dB aproximadamente.

Al final de todas las retransmisiones se dispone de dos vectores que almacenan los valores de SNR y sus correspondientes valores de probabilidad de error. A partir de estos dos vectores será posible la elaboración de una curva de probabilidad de error que sirva para determinar cuáles son los errores existentes en determinados niveles de ruido.

En la mayoría de las gráficas que contienen estas curvas se realiza una comparativa entre sistemas con diferentes características (uso de diferentes moduladores o diferentes tamaños de FFT). Para la realización de las mismas habrá que realizar las 49 retransmisiones en cada uno de los sistemas a comparar y se obtendrán tantas parejas

de vectores con sus correspondientes valores de relación señal a ruido y de probabilidad de error como sistemas se desee contrastar.

La probabilidad de error se expresará en un eje con escala logarítmica en lugar de una escala lineal.

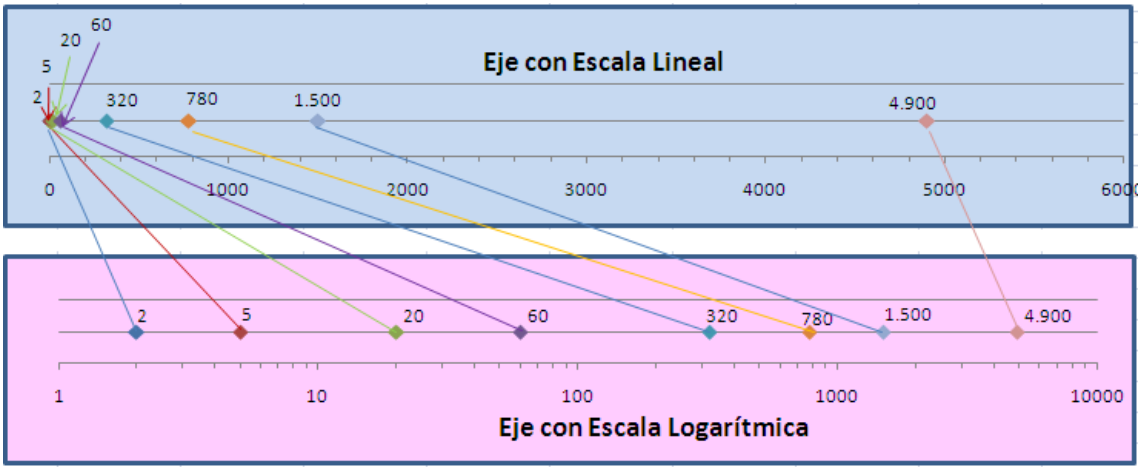

Figura 6.6 *Eje con Escala Logarítmica y con Escala Lineal*

La razón para el uso de esta escala es que permite una mejor visualización y comparación de las variaciones de los valores de la probabilidad de error en un caso con un rango de variación tan grande como éste.

7. VALIDACIÓN

Esta parte de la memoria se corresponde con la consecución de las gráficas que demuestran que el sistema está bien implementado.

En la primera parte se mostrarán las estimaciones del canal con diferentes tamaños de FFT.

En la segunda aparecerá la curva de probabilidad de error del sistema SISO cuando no existe canal en la comunicación.

7.1. Estimación del Canal

A continuación se mostrarán diferentes estimaciones de un canal cuando no existe ruido en la transmisión. Estas gráficas son el resultado de los cálculos realizados por cualquier ecualizador en cualquiera de los sistemas desarrollados. El uso de varias antenas o de distintos moduladores no afecta a la estimación realizada.

El objetivo es comprobar las variaciones en la estimación del canal a medida que aumenta el tamaño de FFT y por tanto también las señales de referencia.

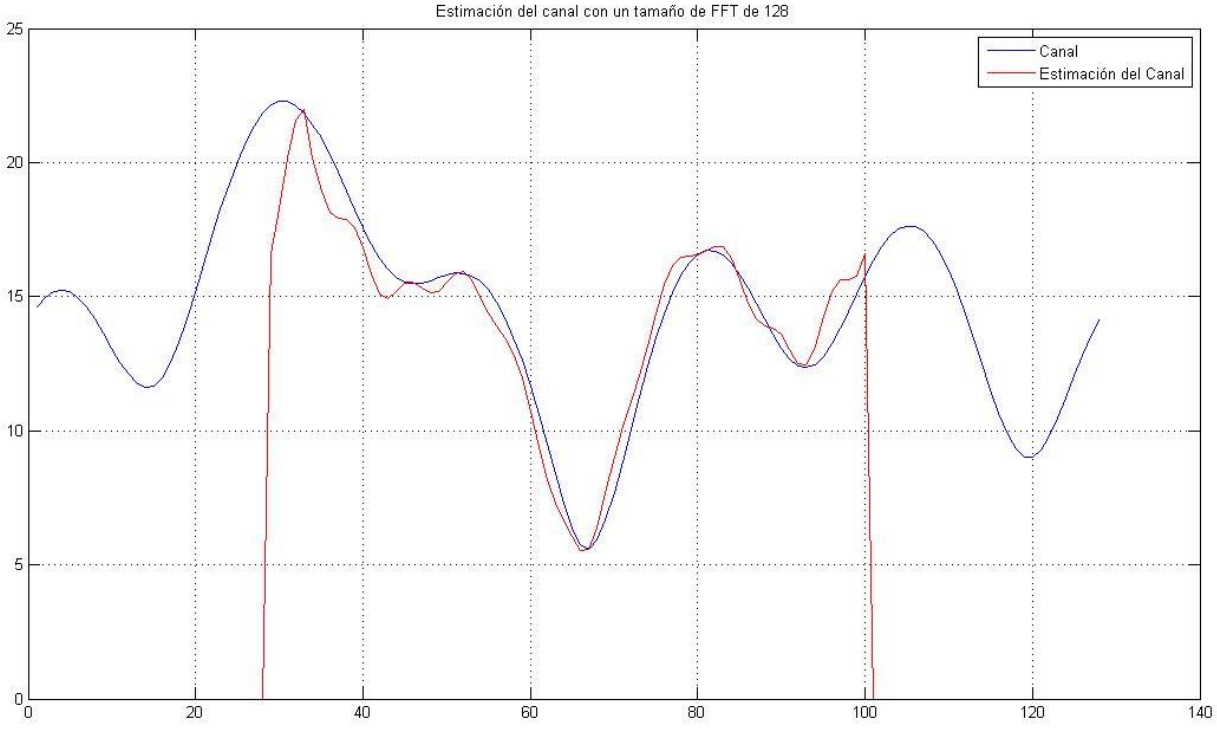

Figura 7.1 *Estimación del canal, modulador QPSK, Nfft = 128*

La anterior estimación realizada se ajusta al canal de forma poco precisa ya que en esta situación el número de señales piloto por símbolo es el menor posible.

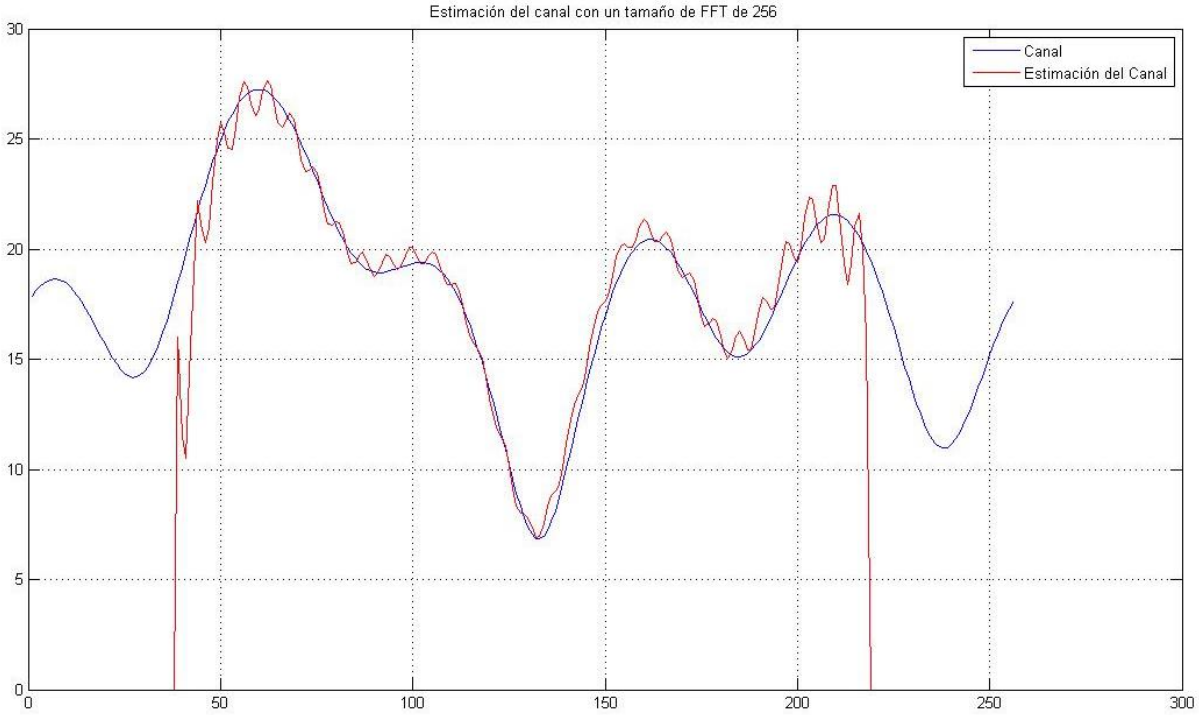

Figura 7.2 *Estimación del canal, modulador QPSK, Nfft = 256*

Con un tamaño de FFT de 256 la estimación mejora. En la imagen anterior se puede apreciar como existe un mayor ajuste de la estimación con respecto al canal.

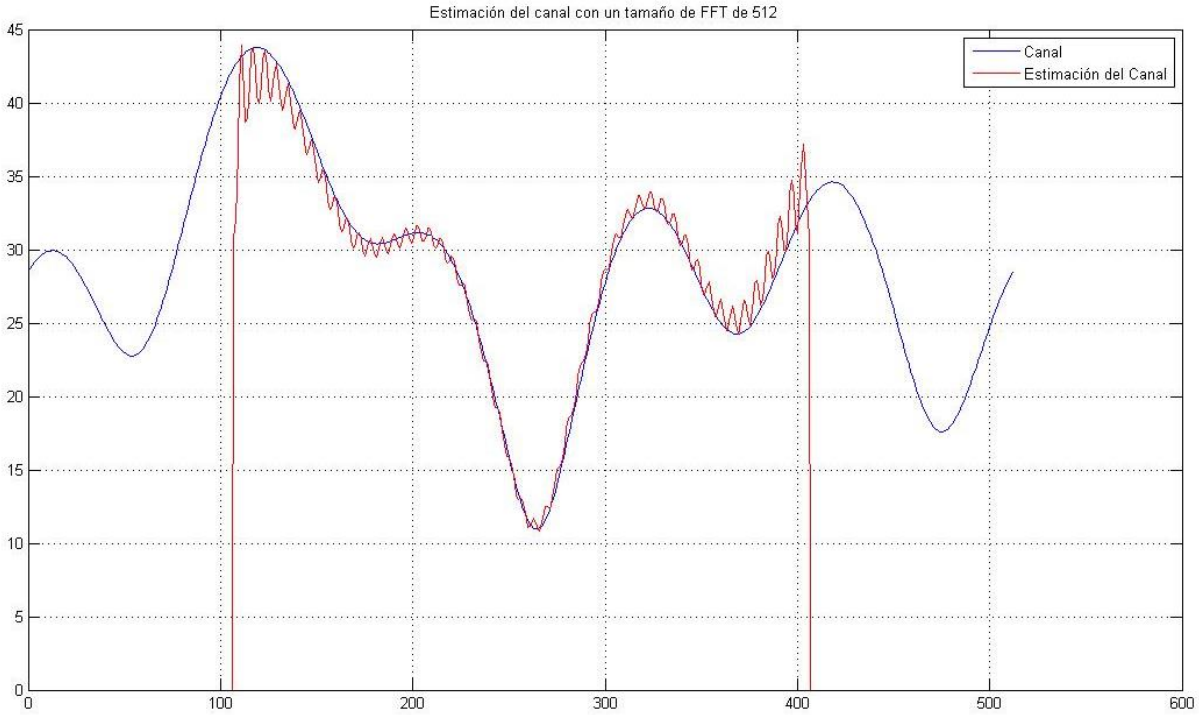

Figura 7.3 *Estimación del canal, modulador QPSK, Nfft = 512*

Para tamaños de FFT de 512 y posteriores, las oscilaciones de la estimación aumentan su frecuencia, produciéndose un acercamiento entre el canal y su estimación.

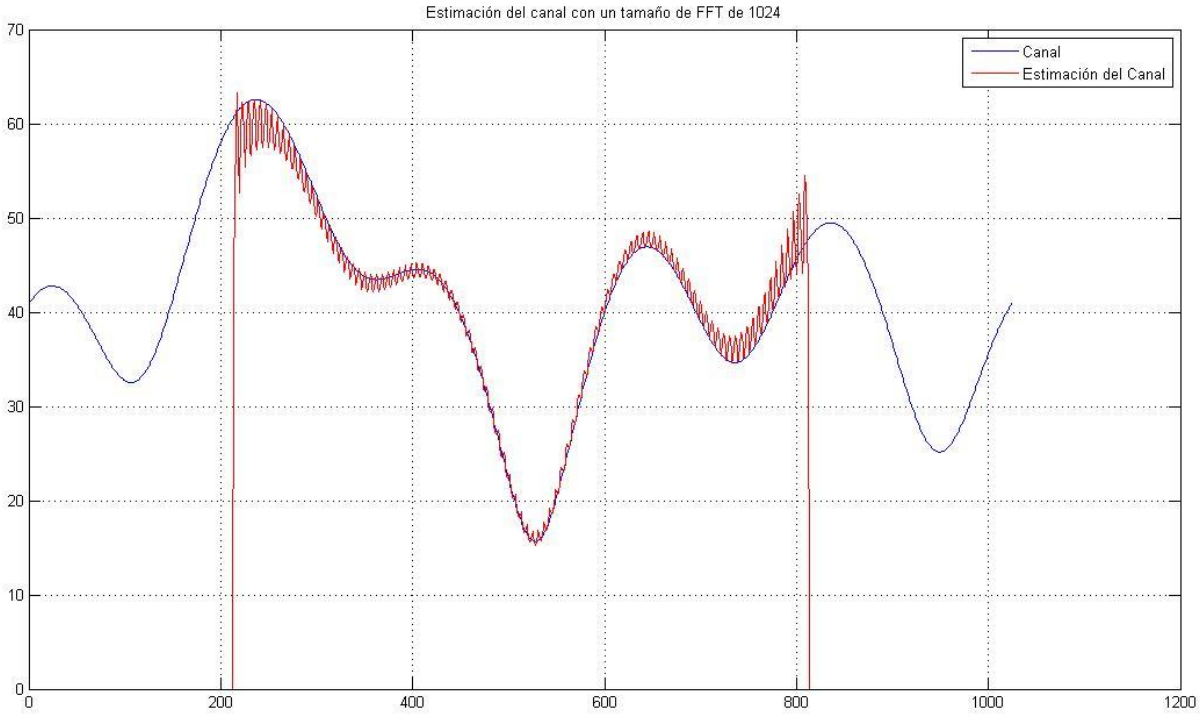

Figura 7.4 *Estimación del canal, modulador QPSK, Nfft = 1024*

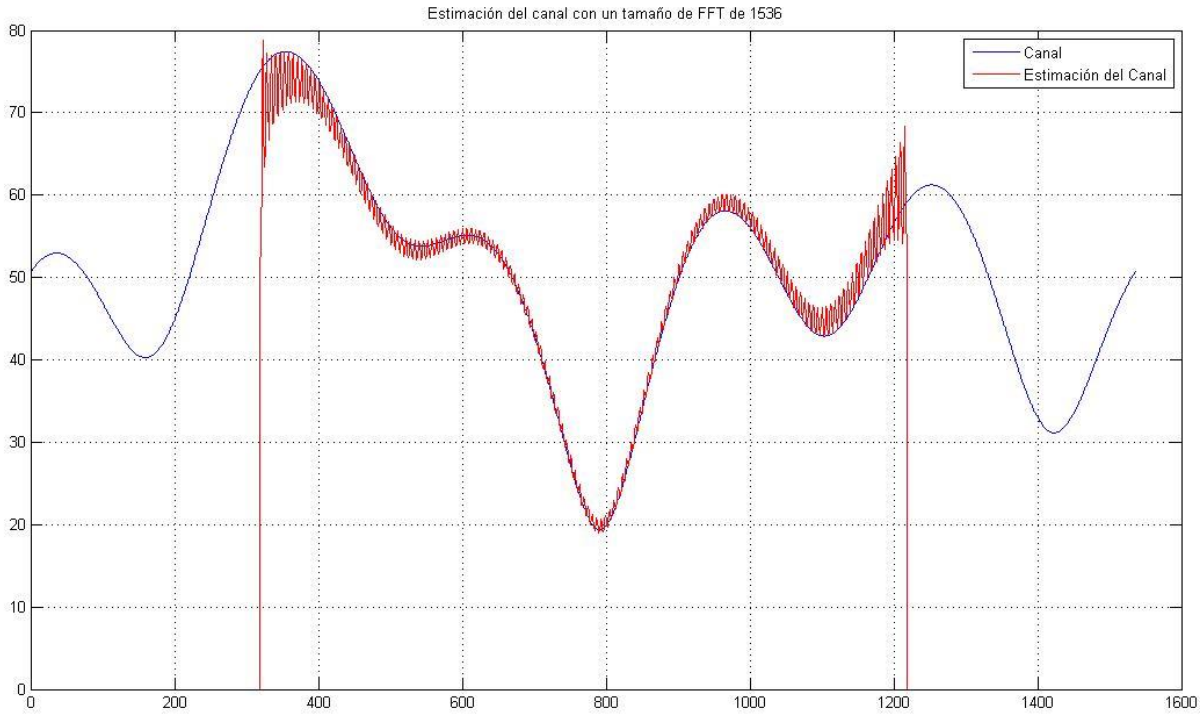

Figura 7.5 *Estimación del canal, modulador QPSK, Nfft = 1536*

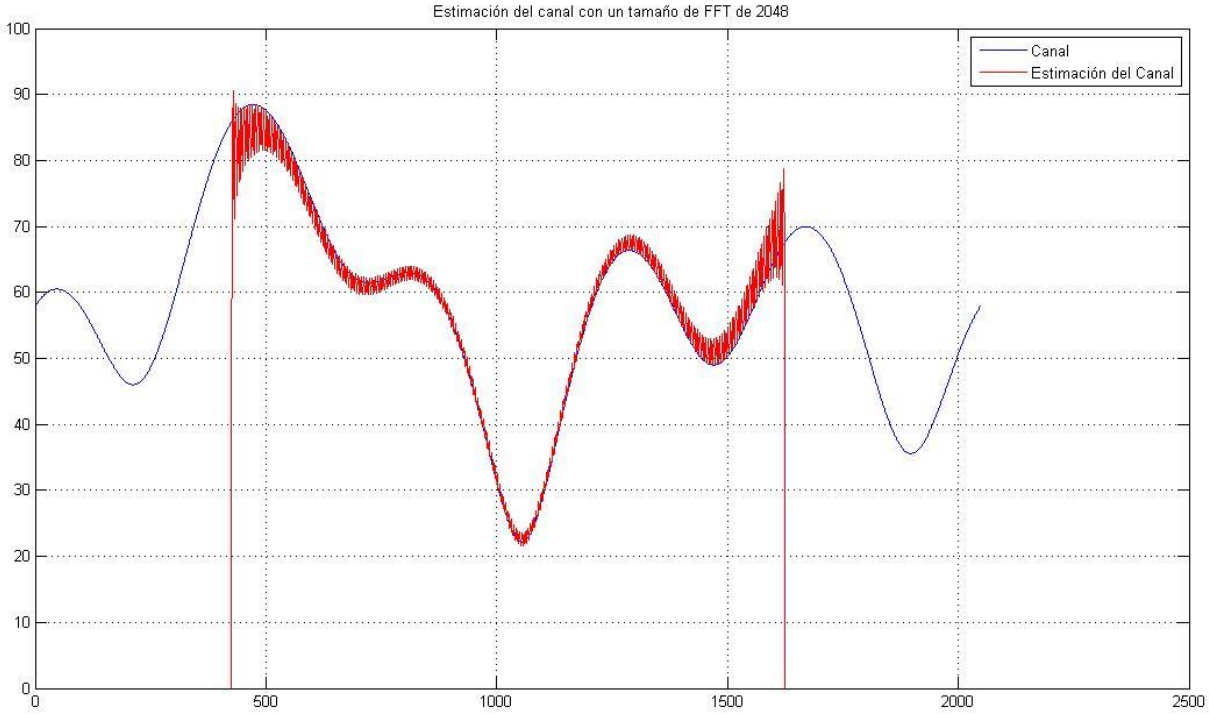

Figura 7.6 *Estimación del canal, modulador QPSK, Nfft = 2048*

Las últimas gráficas de 1024, 1536 y 2048 manifiestan el proceso de mejora de la estimación del canal a partir de un patrón común.

Un mayor tamaño de FFT implica, como se ha introducido anteriormente, un aumento del número de oscilaciones de la estimación en torno al canal. Esta mayor cantidad de oscilaciones provoca que progresivamente la estimación del canal se ajuste cada vez más a la expresión del canal existente.

La mejor estimación posible del canal se produce con un tamaño de FFT de 2048, cuando el número de señales de referencia por símbolo es el máximo posible. En esta situación los cambios de fase y amplitud generados por el canal son corregidos prácticamente en su totalidad gracias a la estimación de canal realizada por el ecualizador.

7.2. Curva de probabilidad de error Sin Canal

La otra parte de la validación consiste en la obtención de la curva de probabilidad de error a diferentes niveles de ruido cuando no existe ningún tipo de canal. Para obtener las gráficas se usan todos los moduladores permitidos en la Release 8 para el enlace de bajada MIMO, es decir, el modulador QPSK, el 16 QAM y el 64 QAM. La curva está realizada con un tamaño de FFT de 512.

Se obtendrá una comparativa de las curvas obtenidas para los diferentes tipos de moduladores. En estas gráficas se podrá apreciar como el modulador que produce más errores es el de 64 QAM, luego el de 16 QAM y el que menos errores genera es el QPSK. La razón del aumento es que al usar un modulador con más bits por símbolo, los símbolos se encuentran más cerca, disminuye la región de decisión y por tanto es más fácil cometer errores en el receptor. Esta desventaja es compensada por un importante aumento en la velocidad de transmisión.

La forma correcta de validar el sistema es comprobar si existen similitudes entre los resultados obtenidos y las curvas de probabilidad de error de los diferentes moduladores, mostrada en la siguiente figura. Esta gráfica, aunque tiene diferentes unidades en el eje de coordenadas para expresar el ruido existente, es una buena referencia para determinar si la curva de validación del sistema se aproxima a las de la figura y el sistema está bien implementado.

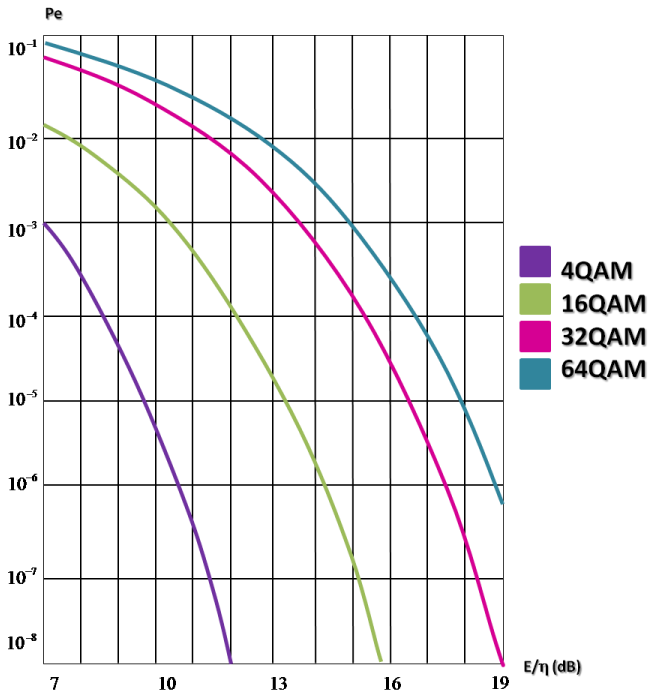

Figura 7.7 *Curva de probabilidad de error de diferentes moduladores*

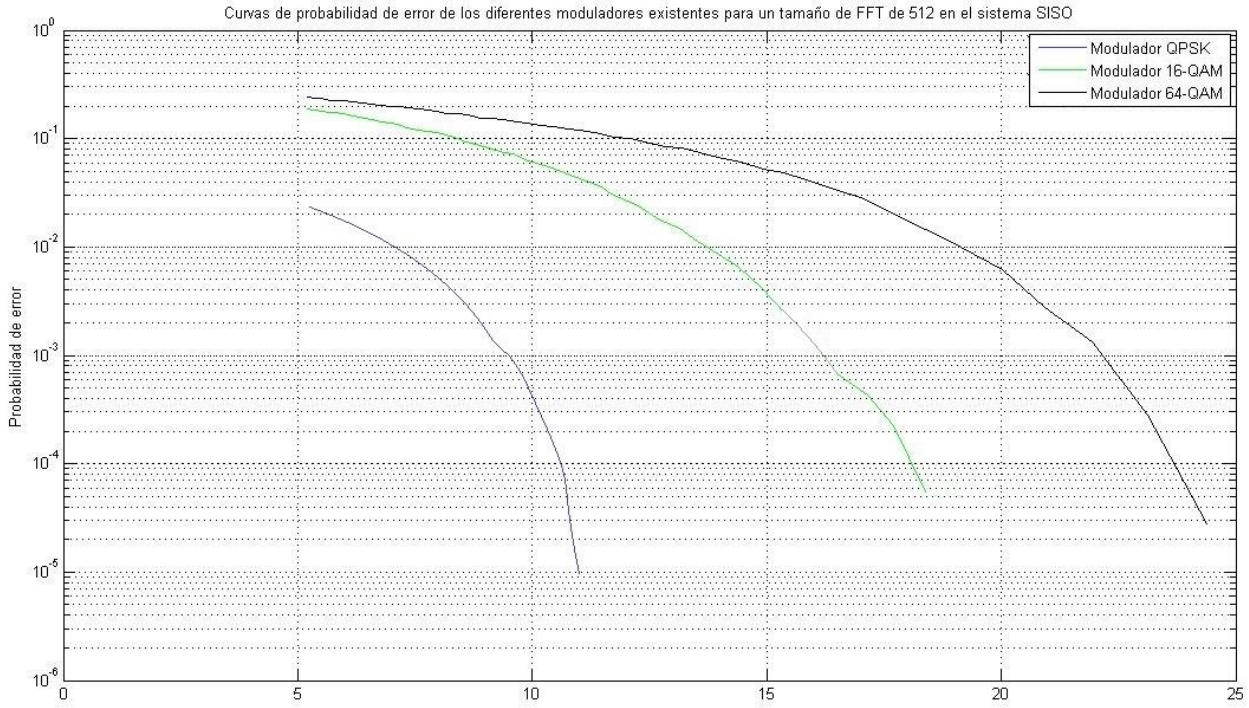

Figura 7.8 *Comparativa de las curvas de probabilidad de error en el sistema SISO sin canal, para los diferentes moduladores, con un tamaño fijo de FFT de 512*

En esta figura se puede observar como las curvas para cada uno de los moduladores poseen una forma similar a las curvas teóricas de la figura 7.7. Los valores del eje de abscisas son diferentes ya que se usan diferentes unidades de ruido.

8. RESULTADOS DE SIMULACIÓN

En esta parte del proyecto se procederá a la realización de las mediciones. Existirá un apartado para cada uno de los sistemas desarrollados en el proyecto (SISO, MIMO 2x2 y MIMO 4x4).

En cada una de los apartados existen nueve gráficas totales cuyo propósito principal es realizar una comparación entre diferentes curvas de probabilidad de error para mostrar las variaciones de los errores existentes en las diferentes implementaciones del sistema con distintos valores de ruido.

El primer tipo de gráficas consistirán en la comparación entre las curvas de probabilidad de error para diferentes tamaños de FFT. Existirán tres gráficas en total, una gráfica para el modulador QPSK, otra para el 16-QAM y otra para el 64-QAM. En cada una de las gráficas aparecerán simultáneamente seis curvas de probabilidad de error para cada uno de los tamaños de FFT (128, 256, 512,1024, 1536 y 2048).

El propósito fundamental será verificar dentro de una misma gráfica el comportamiento del sistema con diferentes tamaños de FFT. Además comparando distintas gráficas, se podrá contrastar el comportamiento general que sufren las curvas de probabilidad de error al cambiar de modulador.

Las seis gráficas restantes se corresponden con la superposición de curvas de probabilidad de error de diferentes moduladores para un tamaño de FFT fijo, es decir, tendremos una gráfica para cada uno de los tamaños de FFT (128, 256, 512,1024, 1536 y 2048). En cada uno de los diferentes tamaños se mostrarán las curvas del modulador QPSK, del 16-QAM y del 64-QAM.

En estas mediciones se pretende demostrar la variación del número de errores de los diferentes moduladores para un mismo tamaño de FFT y la variación sufrida al cambiar ese tamaño.

Como se ha ido explicando a lo largo de la presente memoria el uso de un modulador con un mayor número de bits por símbolo aumenta la velocidad de transmisión y el número de errores.

El modulador 64-QAM consta de 6 bits por símbolo, el 16-QAM de 4 bits por símbolo y el QPSK de 2 bits por símbolo. Es decir, el modulador 64-QAM puede enviar el triple de información que el modulador QPSK en el mismo tiempo, aunque su desventaja es que tiene muchos más errores. El modulador 16-QAM envía el doble de información que el QPSK y aunque no tiene tanta velocidad como el 64-QAM produce menores errores que éste durante la comunicación.

8.1. SISO

Este primer subapartado mostrará varias comparativas de las diferentes curvas de probabilidad de error obtenidas en la comunicación (ya con un canal SCM) para la tecnología SISO.

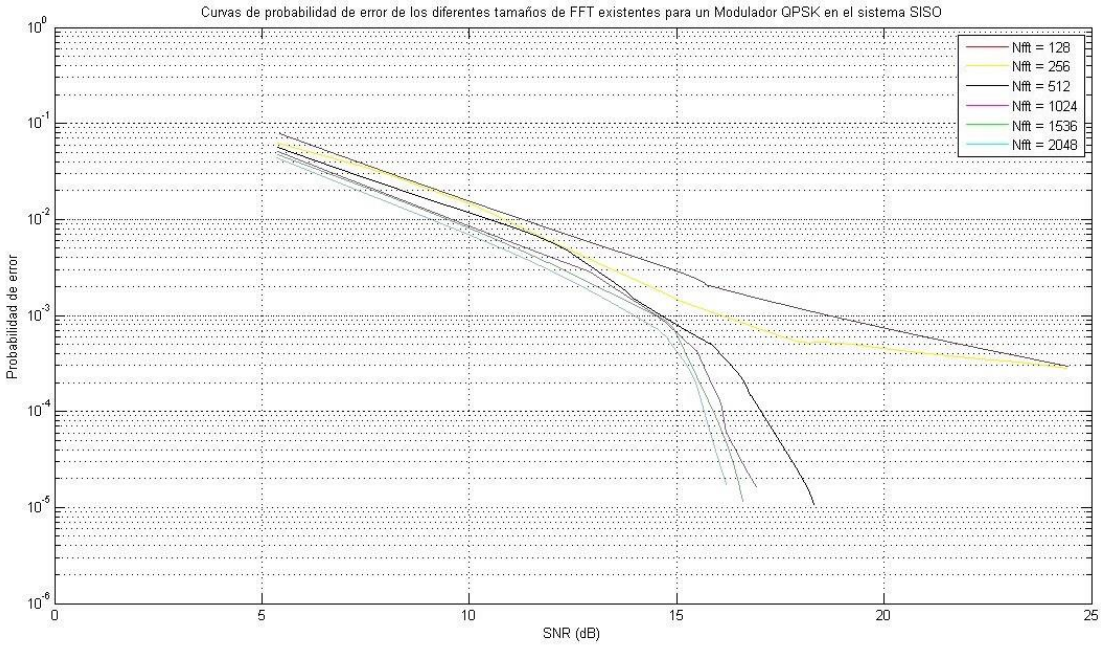

Figura 8.1 *Comparativa de las curvas de probabilidad de error en el sistema SISO con canal, para diferentes tamaños de FFT, con el modulador QPSK*

En esta figura se puede comprobar las diferentes probabilidades de error cuando el sistema SISO usa un modulador QPSK. El modulador QPSK proporciona la menor tasa de errores posible.

Las diferencias de probabilidad de error entre los diferentes tamaños de FFT son debidas a que en los casos con un mayor tamaño de FFT existe una mejor estimación del canal y por tanto una menor probabilidad de error, ya que hay un mayor número de señales piloto que en los casos con menor tamaño de FFT donde los fallos son mayores.

A menores valores de ruido se aprecia la disminución de los errores cuando aumenta el tamaño de FFT. Se puede apreciar una importante diferencia entre los errores existentes de los sistemas con tamaño de 128 y 512 frente al resto.

A medida que aumenta el ruido existente en la comunicación y disminuye la relación señal a ruido, las curvas de probabilidad de error de los diferentes tamaños de FFT tienden a converger, es decir, el aumento del ruido afecta en mayor medida a los

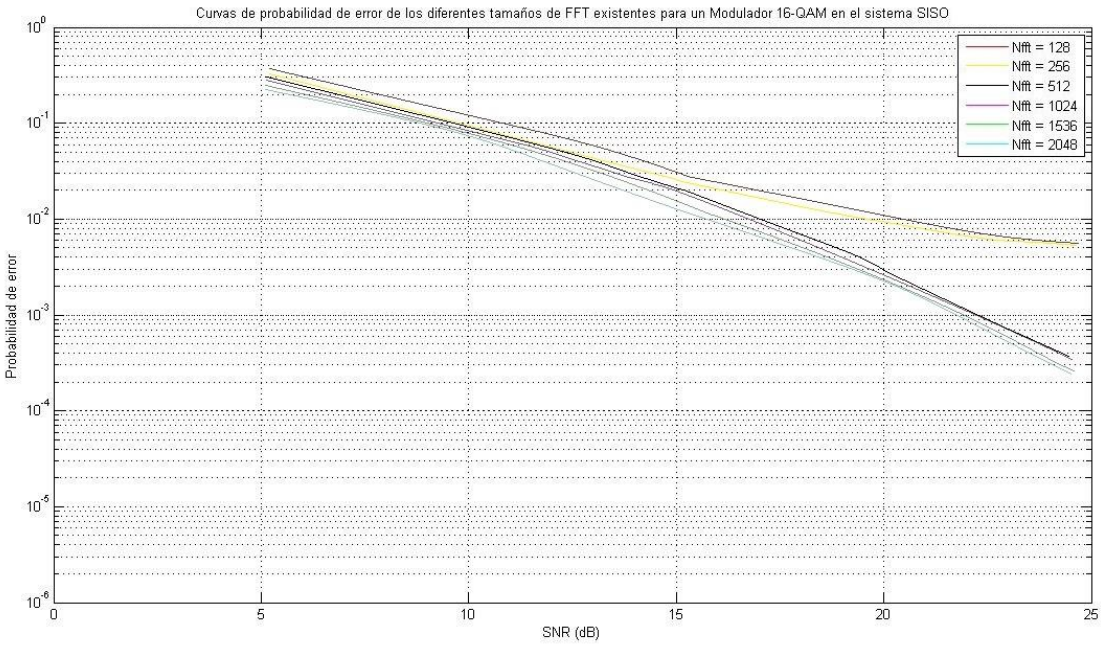

Figura 8.2 *Comparativa de las curvas de probabilidad de error en el sistema SISO con canal, para diferentes tamaños de FFT, con el modulador 16-QAM*

En estos resultados se puede comprobar cómo el comportamiento de las curvas de probabilidad de error sigue una tendencia muy similar a las curvas de la Figura 8.1, en este caso, con un mayor número de errores debido al uso de un modulador 16-QAM.

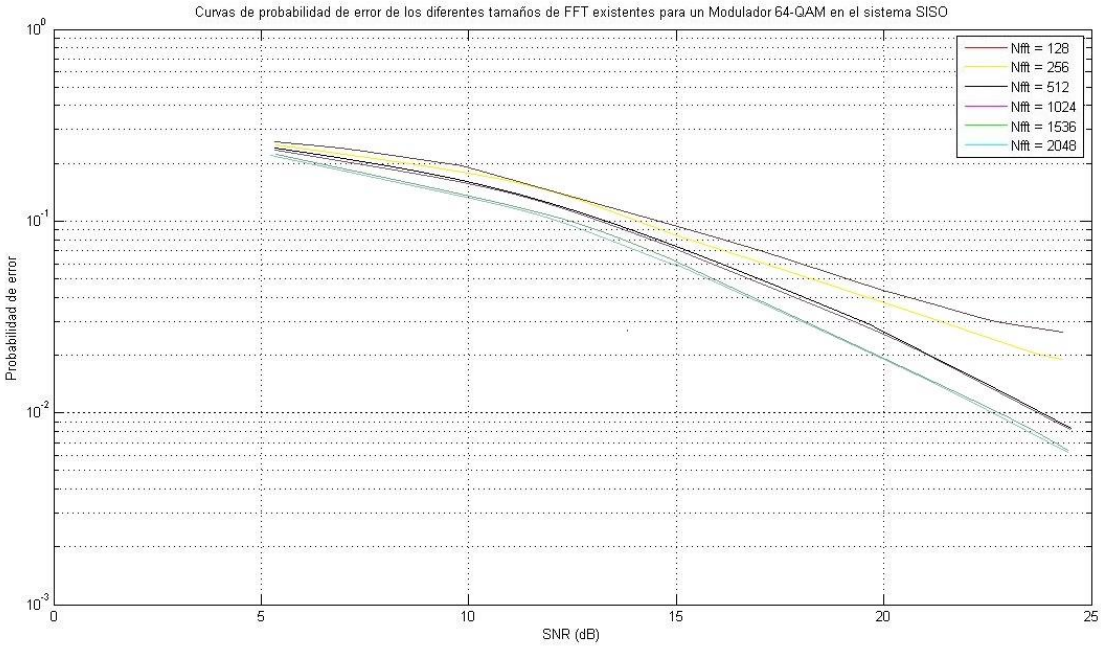

Figura 8.3 *Comparativa de las curvas de probabilidad de error en el sistema SISO con canal, para diferentes tamaños de FFT, con el modulador 64-QAM*

Cuando se usa el modulador 64-QAM el comportamiento de todas las curvas tiende a seguir una pauta muy similar. La variación de las curvas frente a la disminución de la relación señal a ruido es prácticamente idéntica y la diferencia entre los errores para distintos tamaños de FFT es mucho menor que con los otros dos moduladores.

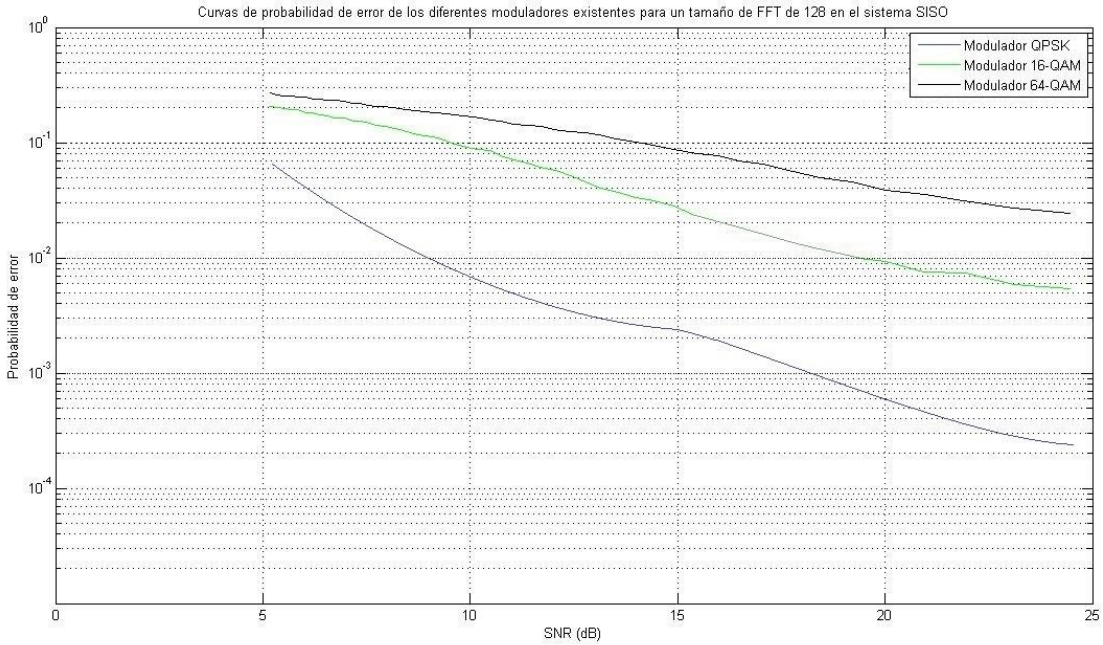

Figura 8.4 *Comparativa de las curvas de probabilidad de error en el sistema SISO con canal, para diferentes moduladores, con tamaño de FFT de 128*

En esta primera figura en la que se realiza una comparativa entre los diferentes tipos de moduladores para un tamaño de FFT constante de 128, se observa la clara diferencia de errores existente entre los moduladores. Esta diferencia es mayor cuando hay una alta relación señal a ruido que cuando la relación es menor.

El modulador con menor número de bits por símbolo, el QPSK, consigue unas tasas de error muy notables, con errores muy bajos. En cambio en los otros moduladores existe un importante aumento de los errores, como consecuencia de la transmisión de más bits en un mismo símbolo lo que conduce a una disminución del tamaño de las regiones de decisión. A pesar de este aumento de errores no se puede pasar por alto que un modulador 16-QAM proporciona el doble de velocidad que el modulador QPSK y que el modulador 64-QAM proporciona el triple de velocidad que el QPSK.

Observando en detalle cada curva, se hace evidente una tendencia que será repetida en las siguientes gráficas. La curva de probabilidad de error del modulador QPSK sufre un mayor aumento de errores cuando aumenta el nivel de ruido que las demás. Es decir, cuando el modulador dispone de un mayor número de bits por símbolo, en general posee mayores tasas de probabilidad de error, pero la curva es más estable, los errores aumentan en una menor proporción a medida que en la comunicación existen mayores niveles de ruido.

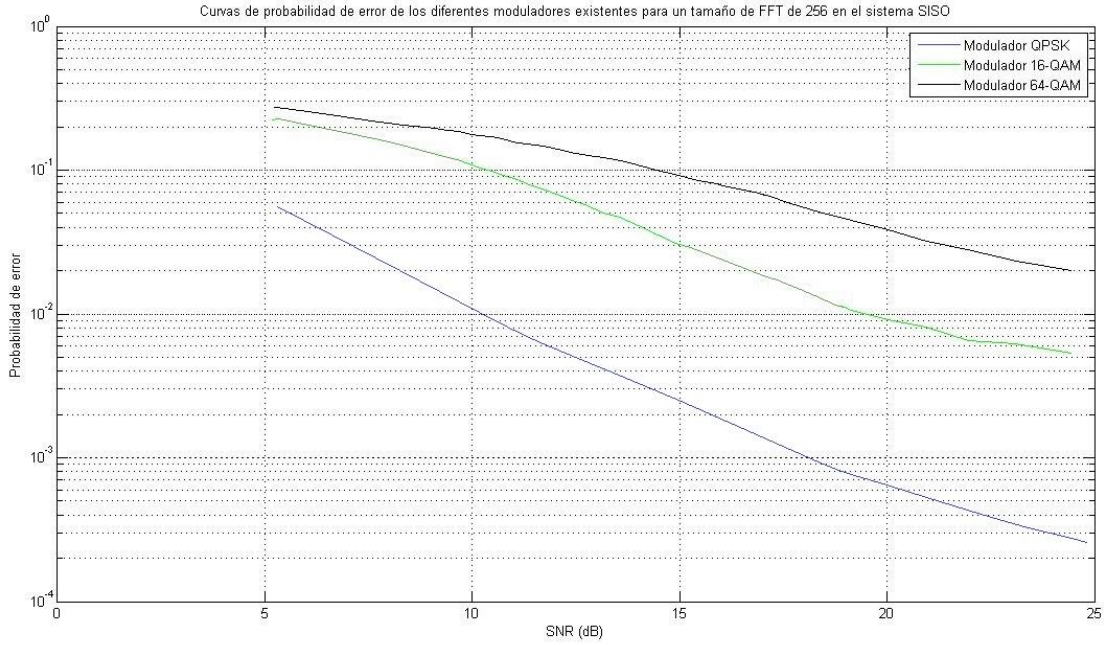

Figura 8.5 *Comparativa de las curvas de probabilidad de error en el sistema SISO con canal, para diferentes moduladores, con tamaño de FFT de 256*

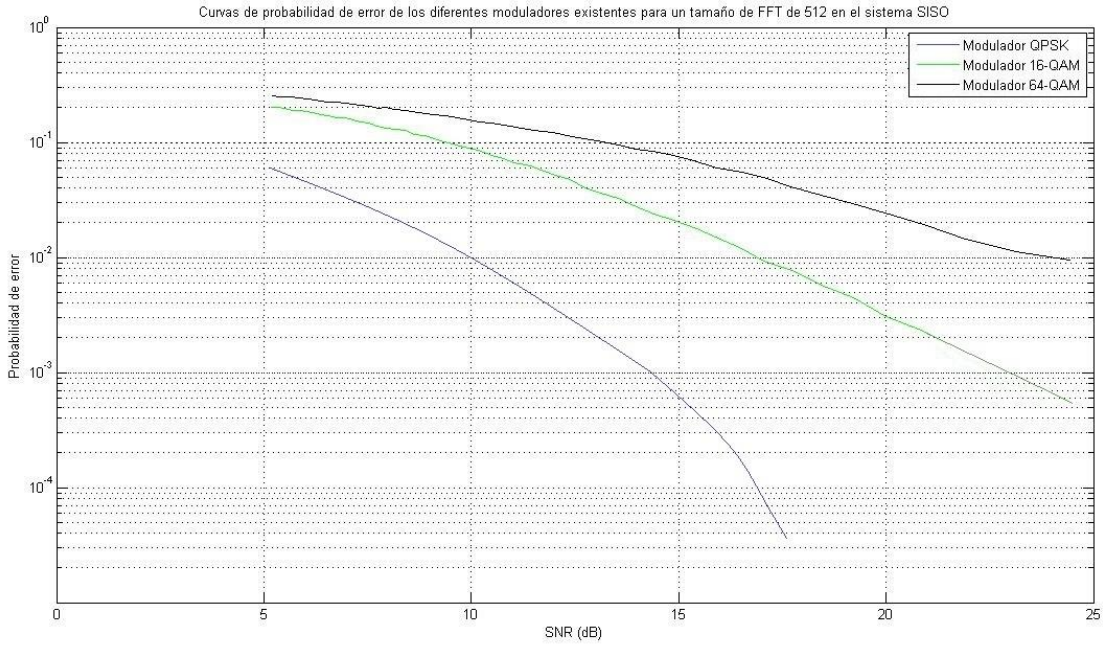

Figura 8.6 *Comparativa de las curvas de probabilidad de error en el sistema SISO con canal, para diferentes moduladores, con tamaño de FFT de 512*

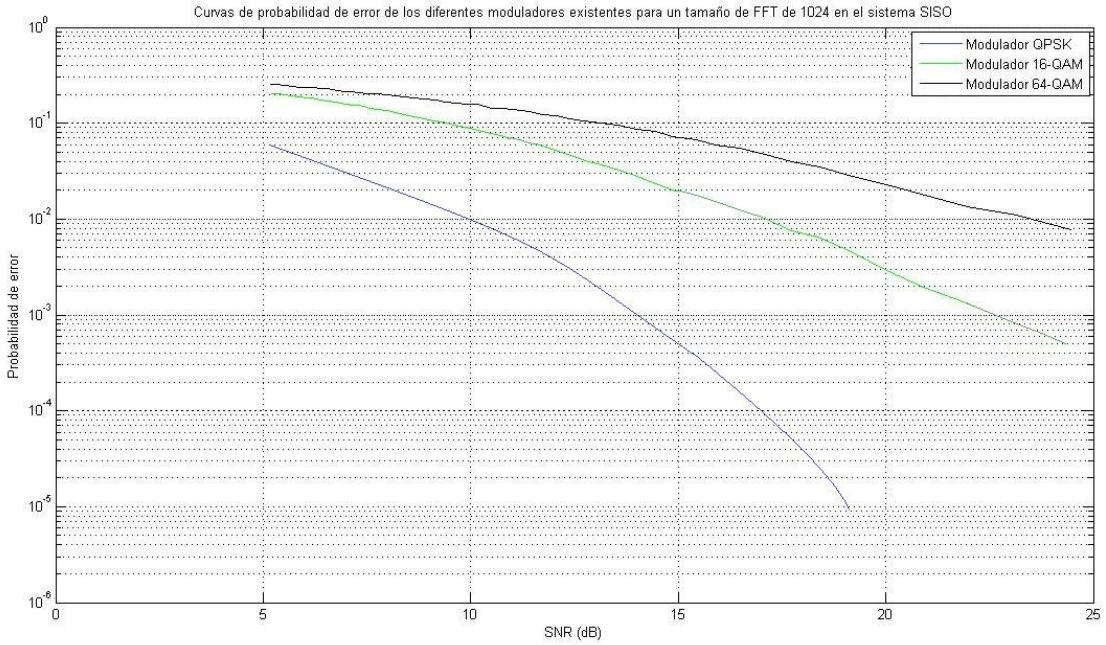

Figura 8.7 *Comparativa de las curvas de probabilidad de error en el sistema SISO con canal, para diferentes moduladores, con tamaño de FFT de 1024*

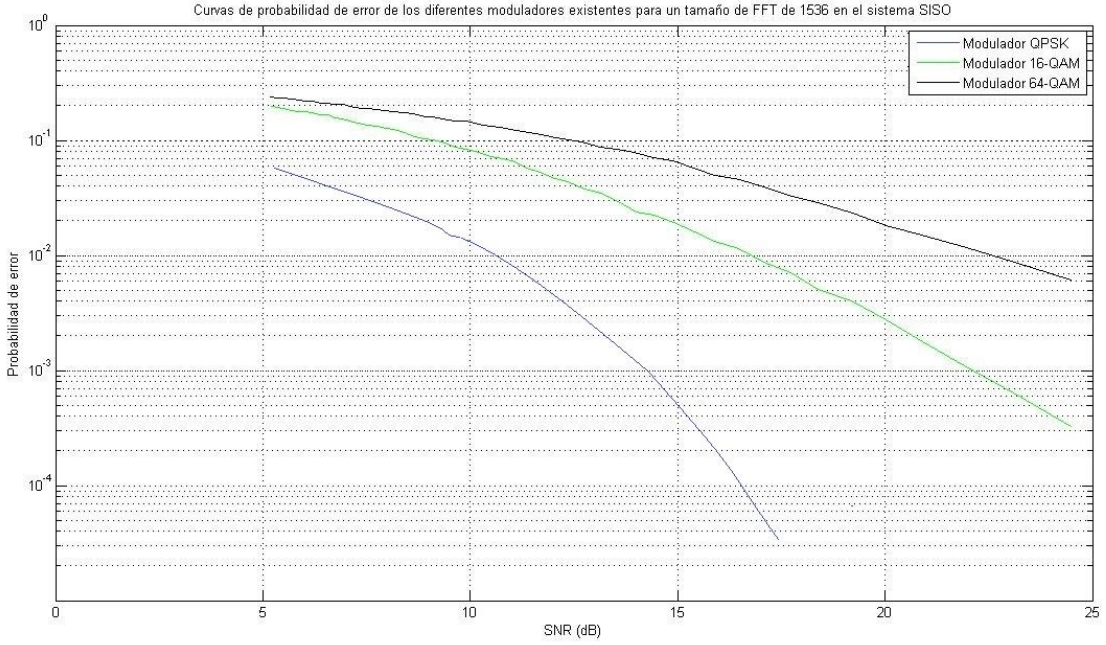

Figura 8.8 *Comparativa de las curvas de probabilidad de error en el sistema SISO con canal, para diferentes moduladores, con tamaño de FFT de 1536*

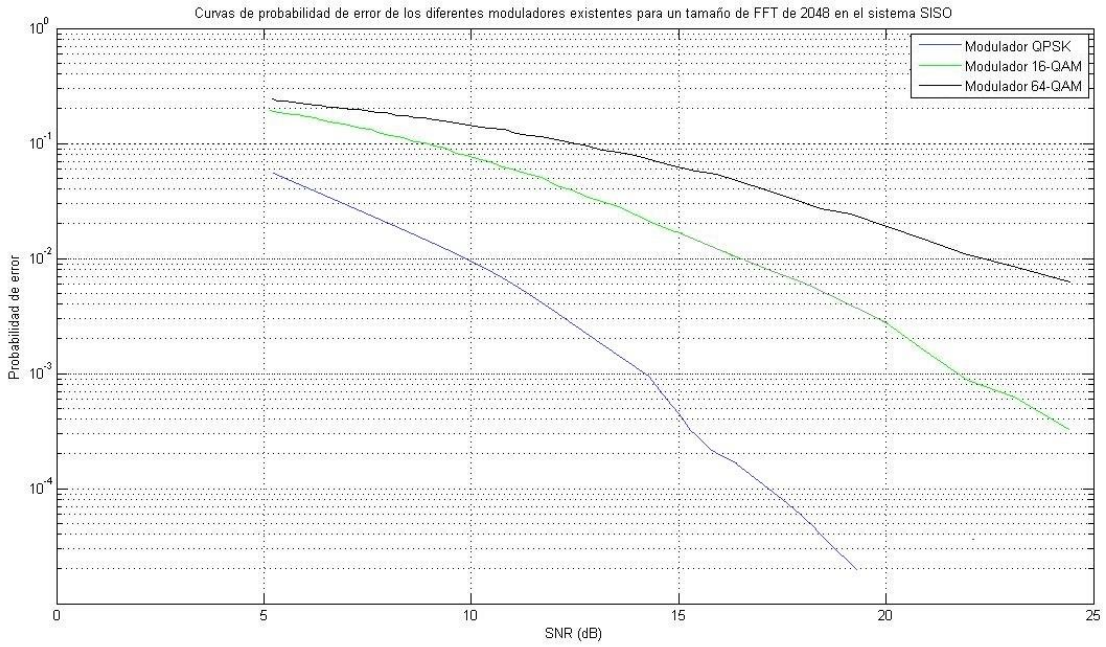

Figura 8.9 *Comparativa de las curvas de probabilidad de error en el sistema SISO con canal, para diferentes moduladores, con tamaño de FFT de 2048*

En las figuras 8.5, 8.6, 8.7, 8.8 y 8.9 se puede apreciar como las diferencias entre los valores de probabilidad de error de los moduladores QPSK, 16-QAM y 64-QAM se mantienen constantes a medida que aumenta el tamaño de FFT siguiendo un comportamiento similar a la primera gráfica comentada (8.4). Es normal que la diferencia de errores entre los moduladores sea menor cuando hay más ruido en el canal que cuando hay menos.

Es perceptible el hecho de que a medida que aumenta el tamaño de FFT se puede comprobar cómo el porcentaje de errores existente en la comunicación disminuye para cualquiera de los moduladores.

Este descenso de los fallos es fácilmente apreciable cuando el nivel de ruido en el canal es bajo. Sin embargo, los valores a mayor nivel de ruido son muy similares unos a otros, siendo complicado detectar el ligero descenso de los errores al aumentar el tamaño de FFT cuando la relación señal a ruido es baja.

Tras estas primeras gráficas de simulación es remarcable mencionar la cantidad de matices existentes entre las curvas de probabilidad de error, ya sea modificando el tamaño de FFT o el modulador usado y sobre todo viendo los diferentes comportamientos entre diferentes niveles de ruido.

En los posteriores subapartados aparecerán las gráficas de MIMO 2x2 y MIMO 4x4, lo que servirá no sólo para comprobar las variaciones existentes en cada una de las dos tecnologías al modificar sus características fundamentales, sino también para observar cómo afecta la adición de antenas a la comunicación y si las pautas de las curvas de probabilidad de error varían.

8.2. MIMO 2x2

En este punto se tratará la tecnología MIMO con dos antenas transmisoras y dos antenas receptoras.

Las gráficas mostrarán los errores que se producen en el sistema con diferentes valores de ruido. Los fallos consistirán en un promedio de los errores producidos en cada una de las dos antenas del sistema.

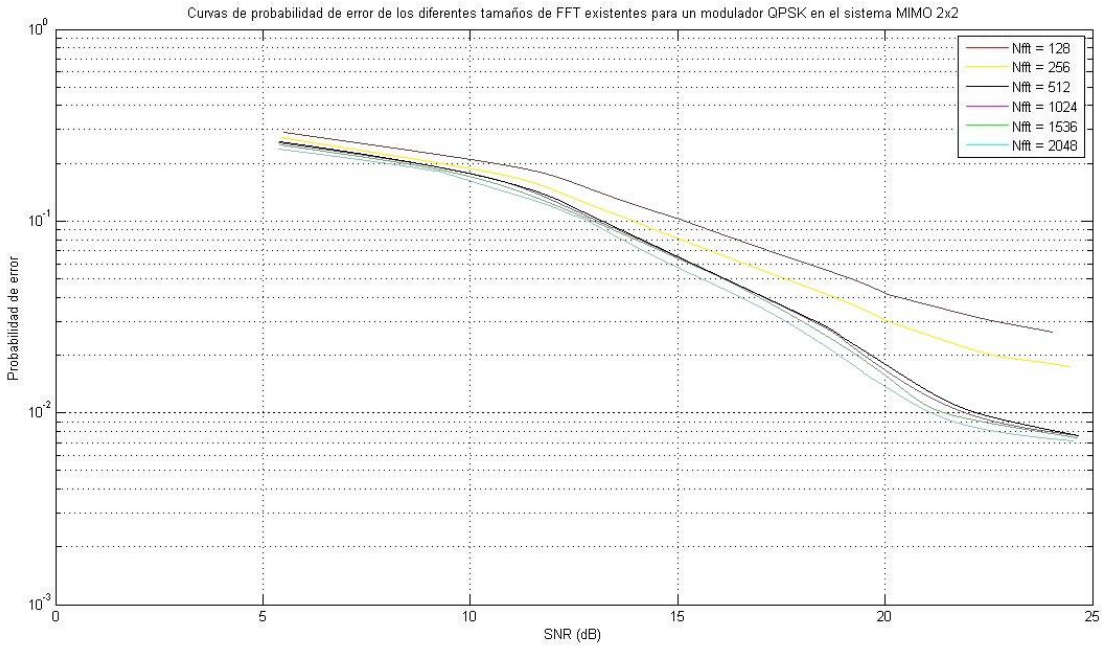

Figura 8.10 *Comparativa de las curvas de probabilidad de error en el sistema MIMO 2x2 con canal, para diferentes tamaños de FFT, con el modulador QPSK*

En esta primera figura del sistema MIMO con 2 antenas se puede constatar como la probabilidad de error de esta tecnología es mayor que en el caso de una única antena. Este aumento es la mayor desventaja con respecto a una tecnología que proporciona el doble de velocidad de transmisión y una mayor robustez frente al aumento del ruido existente, el número de fallos continua siendo mayor con esta tecnología que respecto al caso del sistema SISO, pero el aumento de los errores a medida que disminuye la relación señal a ruido es más leve que en el caso de una única antena transmisora y receptora.

Otro aspecto a destacar es el hecho de que las curvas de probabilidad de error para los diferentes tamaños de FFT están más próximas. En el caso de aquellos tamaños de FFT más altos y que proporcionan un menor número de fallos, sus valores de probabilidad de error son muy similares entre sí.

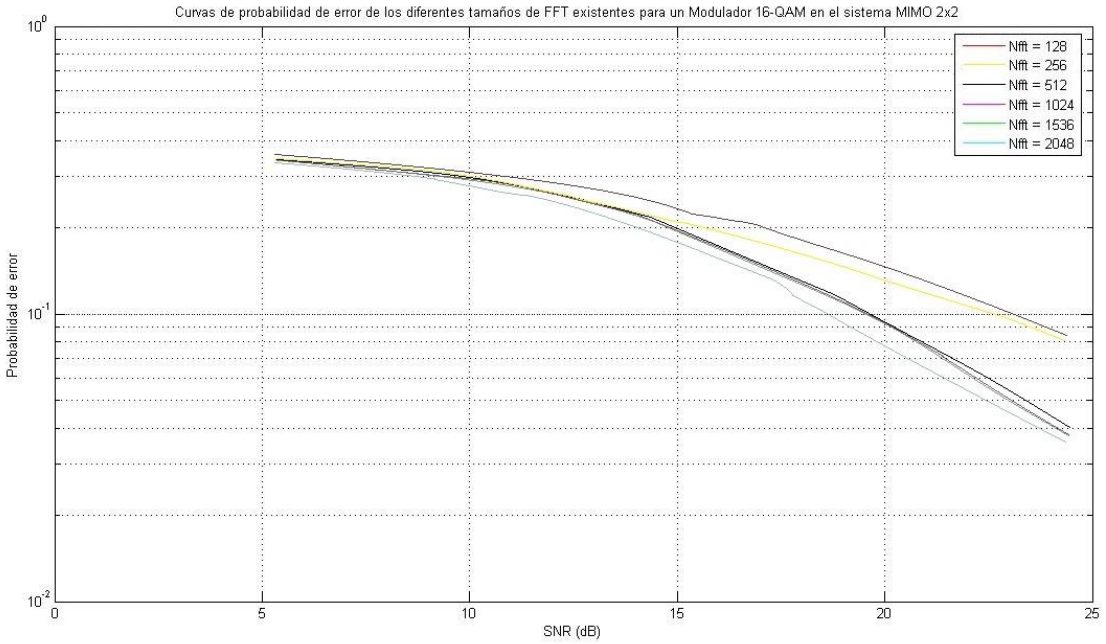

Figura 8.11 *Comparativa de las curvas de probabilidad de error en el sistema MIMO 2x2 con canal, para diferentes tamaños de FFT, con el modulador 16 QAM*

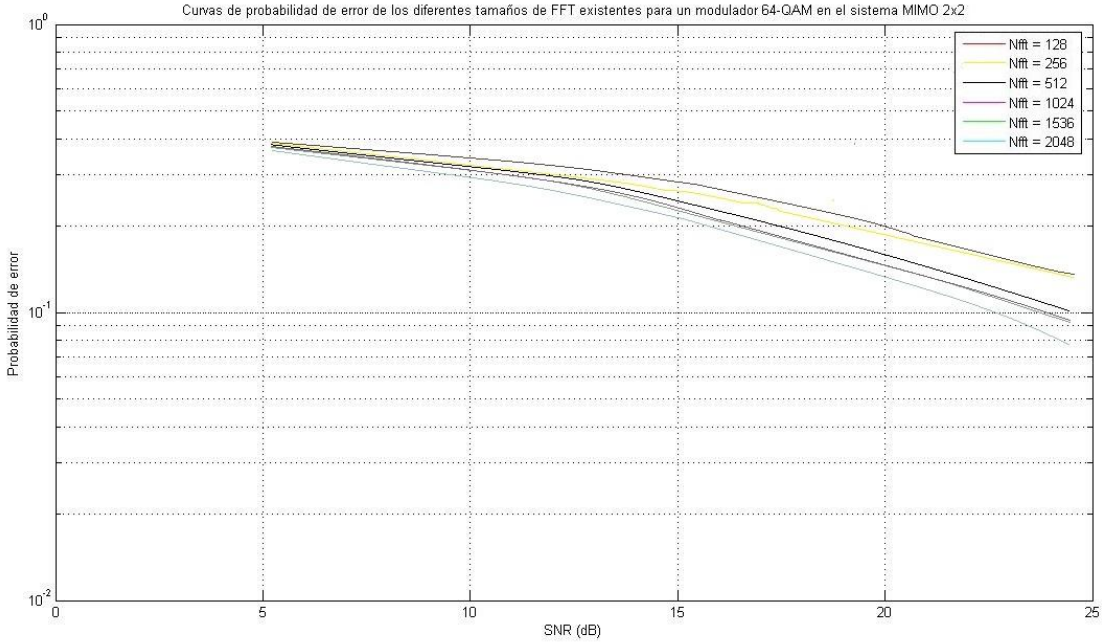

Figura 8.12 *Comparativa de las curvas de probabilidad de error en el sistema MIMO 2x2 con canal, para diferentes tamaños de FFT, con el modulador 64 QAM*

La tendencia del resto de comparativas entre las curvas de los diferentes tamaños de FFT para distintos moduladores continúa con el mismo patrón que en el caso del modulador QPSK.

Las curvas se aproximan cada vez más entre ellas a medida que se usa un modulador con mayor número de bits por símbolo y el aumento de los errores cuando existe mayor cantidad de ruido en el canal cada vez es menor.

Como ya se explicó con anterioridad, el que la probabilidad de error sea menor en los casos que tienen un mayor tamaño de FFT se debe a que la estimación del canal es mejor que cuando el tamaño de FFT es más pequeño.

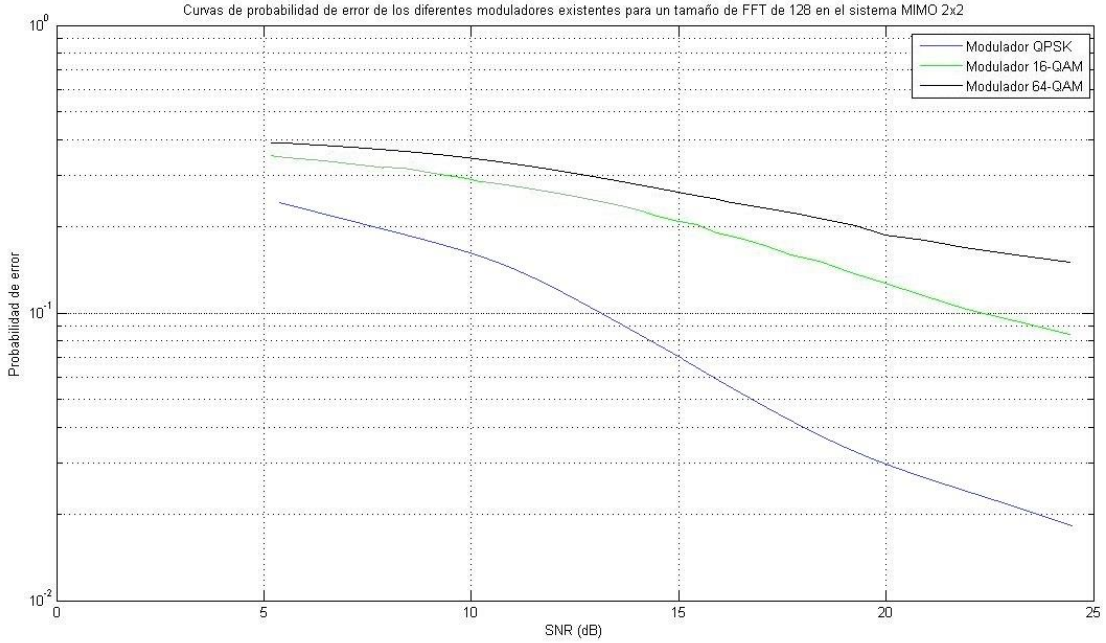

Figura 8.13 *Comparativa de las curvas de probabilidad de error en el sistema MIMO 2x2 con canal, para diferentes moduladores, con tamaño de FFT de 128*

El contraste entre los diferentes moduladores usados en el enlace descendente, muestra que las importantes diferencias de fallos existentes entre cada uno de ellos se mantienen, es decir, en esta tecnología que usa dos antenas en la transmisión y otras dos en la recepción sigue habiendo una diferencia considerable entre la probabilidad de error obtenida para cada uno de los moduladores.

Es muy importante tener claro el tipo de comunicación requerida para poder usar un modulador u otro, teniendo en cuenta algo muy repetido en la presente memoria, el número de errores y la velocidad de transmisión aumentan proporcionalmente al número de bits por símbolo del modulador usado.

Debido al uso de un tamaño de FFT de 128, el número de errores es mayor ya que existe un menor número de señales referencia para estimar el canal.

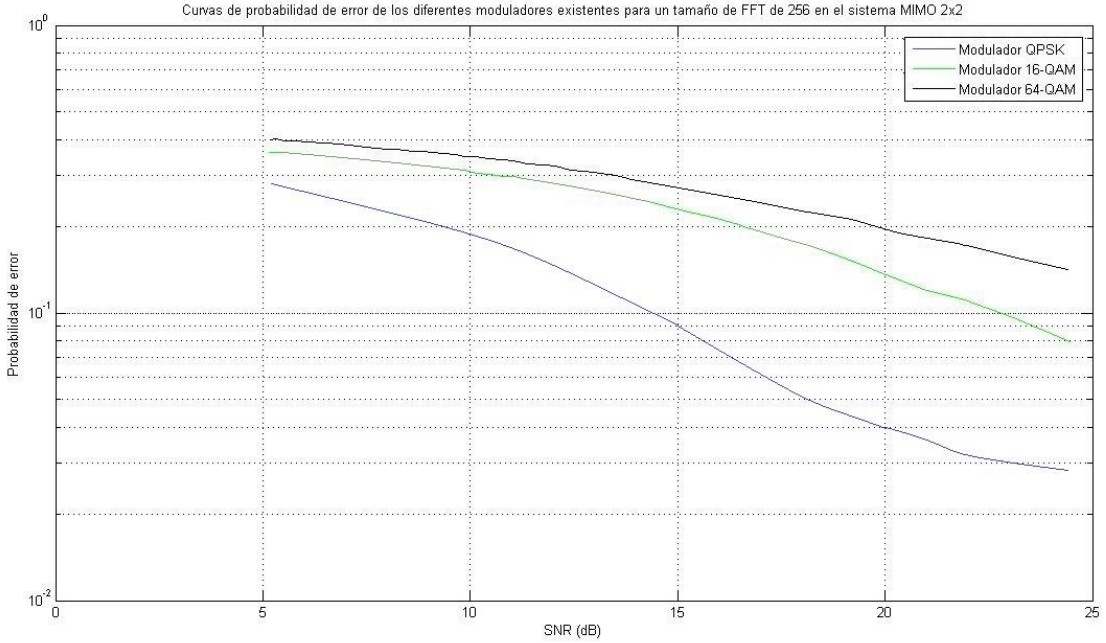

Figura 8.14 *Comparativa de las curvas de probabilidad de error en el sistema MIMO 2x2 con canal, para diferentes moduladores, con tamaño de FFT de 256*

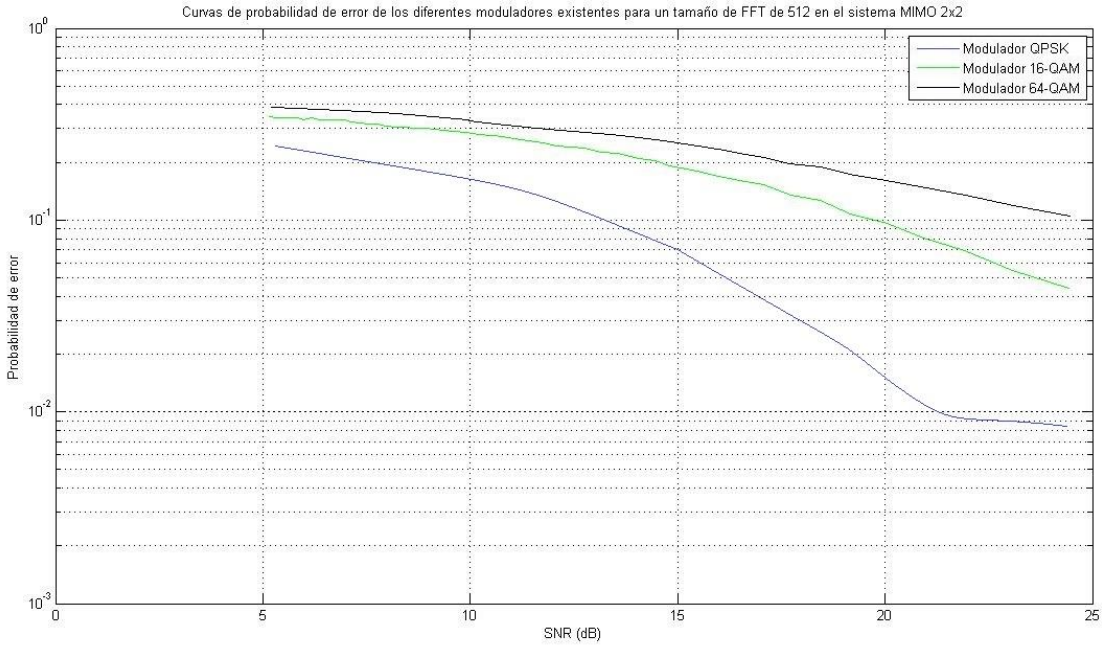

Figura 8.15 *Comparativa de las curvas de probabilidad de error en el sistema MIMO 2x2 con canal, para diferentes moduladores, con tamaño de FFT de 512*

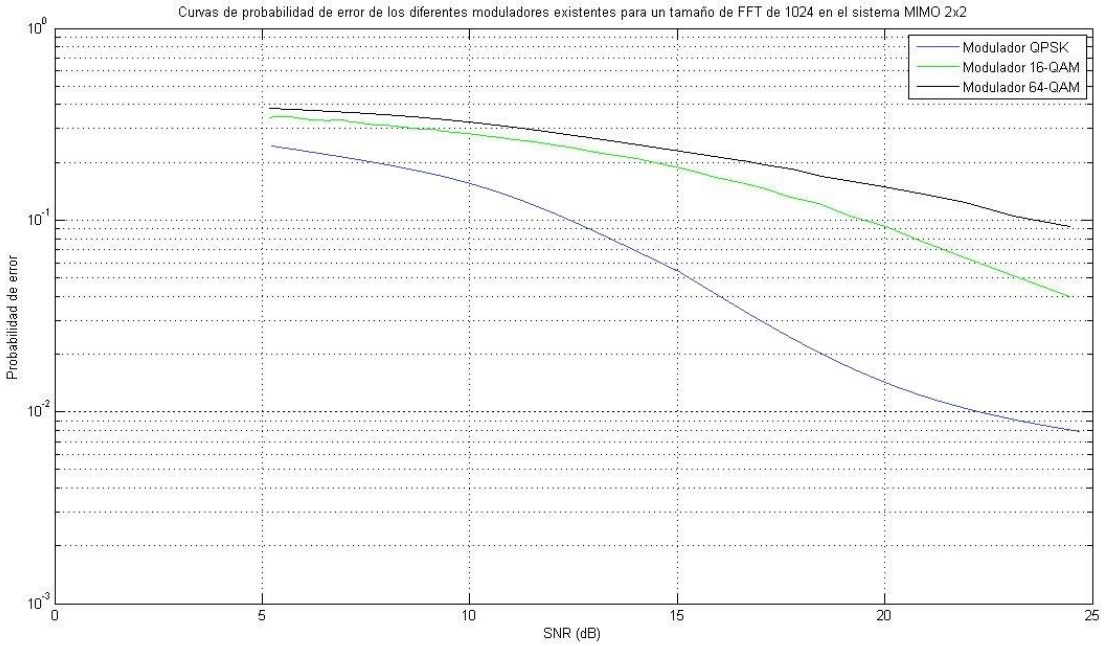

Figura 8.16 *Comparativa de las curvas de probabilidad de error en el sistema MIMO 2x2 con canal, para diferentes moduladores, con tamaño de FFT de 1024*

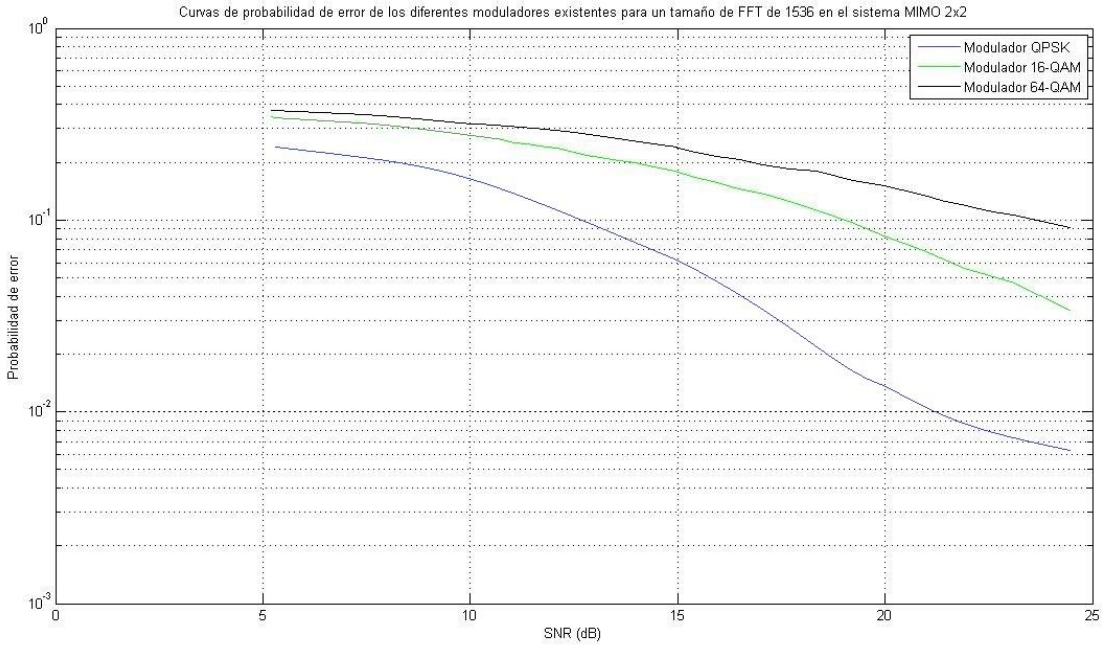

Figura 8.17 *Comparativa de las curvas de probabilidad de error en el sistema MIMO 2x2 con canal, para diferentes moduladores, con tamaño de FFT de 1536*

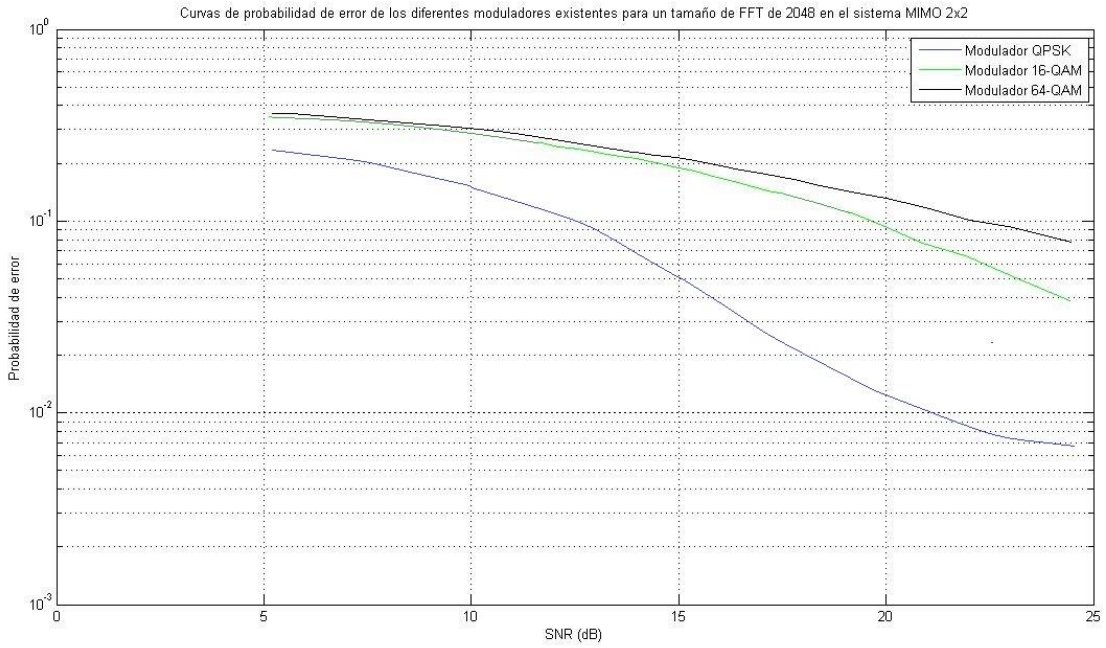

Figura 8.18 *Comparativa de las curvas de probabilidad de error en el sistema MIMO 2x2 con canal, para diferentes moduladores, con tamaño de FFT de 2048*

Las gráficas obtenidas para los tamaños de FFT de 128, 256, 512, 1024, 1536 y 2048 muestran un paulatino descenso de la probabilidad de error. La probabilidad de error sigue siendo mayor que en las curvas del apartado 8.1 de SISO.

A diferencia de las gráficas SISO no existe una diferencia importante entre las curvas de probabilidad de error a medida que aumenta el tamaño de FFT. Es decir, tal y como se ha expuesto en las figuras 8.10, 8.11 y 8.12, el uso de más señales piloto o de referencia no tiene un papel tan trascendente como en el caso de una única antena.

Las curvas cada vez son más parecidas entre sí, tanto en el rango de valores de la probabilidad de error como en la forma de las mismas, su comportamiento frente a un aumento del nivel de ruido es similar en la mayoría de los casos. Estas curvas disponen de una menor pendiente a medida que disminuye la relación señal a ruido, como se ha insistido en la figura 8.10 al señalar una de las ventajas de esta tecnología (el uso de dos antenas receptoras y dos transmisoras implica que el sistema es más robusto frente al aumento de ruido que con el uso de una única antena). Los errores totales con altos niveles de ruido son mayores que en el caso de una única antena, pero el aumento de los mismos respecto a las cotas que disponen de menos ruido en la transmisión es bastante menor.

8.3. MIMO 4x4

Este subapartado consistirá en la simulación del sistema MIMO 4x4.

Se mostrarán las diferentes curvas de probabilidad de error que explicarán los fallos en el sistema. Los fallos totales se obtendrán mediante el promedio de los fallos existentes en cada una de las cuatro antenas.

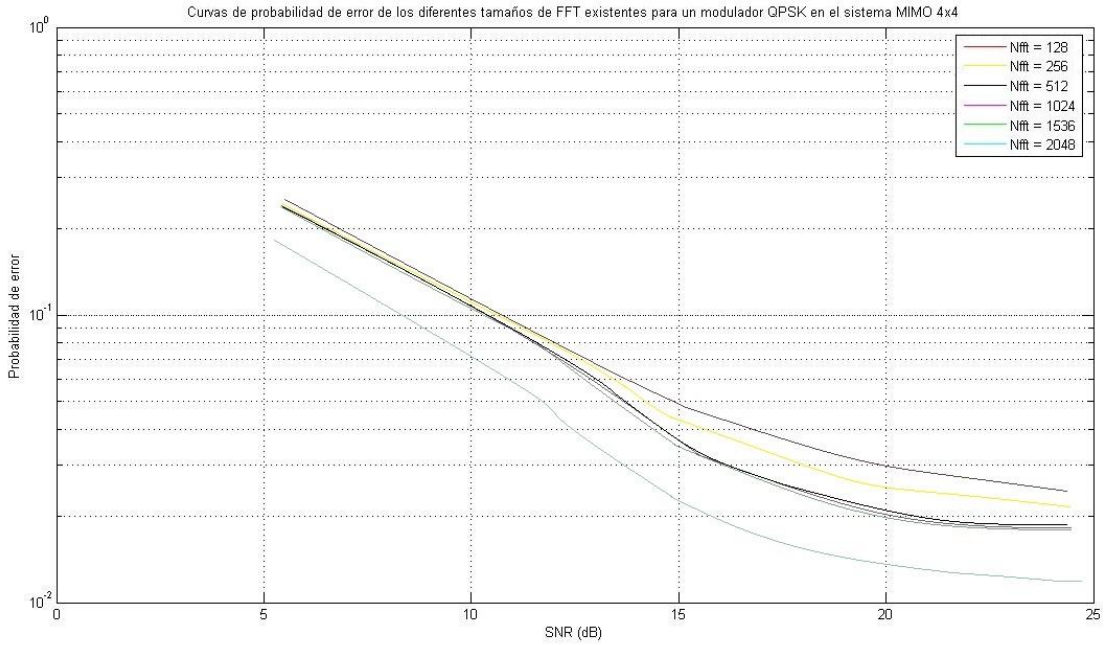

Figura 8.19 *Comparativa de las curvas de probabilidad de error en el sistema MIMO 4x4 con canal, para diferentes tamaños de FFT, con el modulador QPSK*

Las curvas de probabilidad de error en el sistema MIMO 4x4 son muy similares a las curvas obtenidas en MIMO 2x2, los fallos existentes para el modulador QPSK son muy parecidos a los obtenidos en el anterior sistema. Los errores continuan siendo mayores que en la tecnología SISO.

Con un menor tamaño de FFT los errores en MIMO 4x4 son menores que en el anterior sistema mientras que con un tamaño de FFT alto son mayores, siempre y cuando la relación señal a ruido sea alta.

El que haya el doble de antenas confiere al sistema una mayor robusted a medida que aumenta el ruido, la pendiente de la curva tiene una menor inclinación lo que conlleva a que el aumento de los fallos a medida que aumenta el ruido del sistema sea menor. A mayor ruido la probabilidad de error es ligeramente inferior con respecto al sistema MIMO 2x2. MIMO 4x4 tiene el doble de velocidad que MIMO 2x2 sin sufrir un aumento de errores (salvo alguna ligera subida como se ha explicado antes).

Los errores al igual que en casos anteriores siempre son menores en las curvas con mayor tamaño de FFT, debido a que existe un mayor número de señales de referencia y la estimación del canal es mejor.

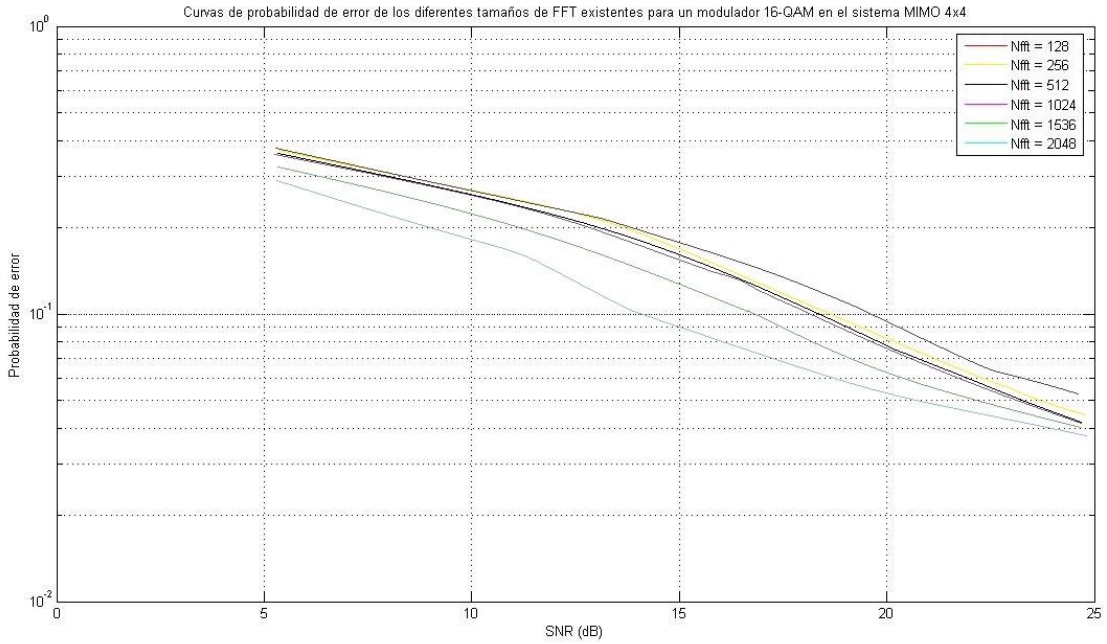

Figura 8.20 *Comparativa de las curvas de probabilidad de error en el sistema MIMO 4x4 con canal, para diferentes tamaños de FFT, con el modulador 16 QAM*

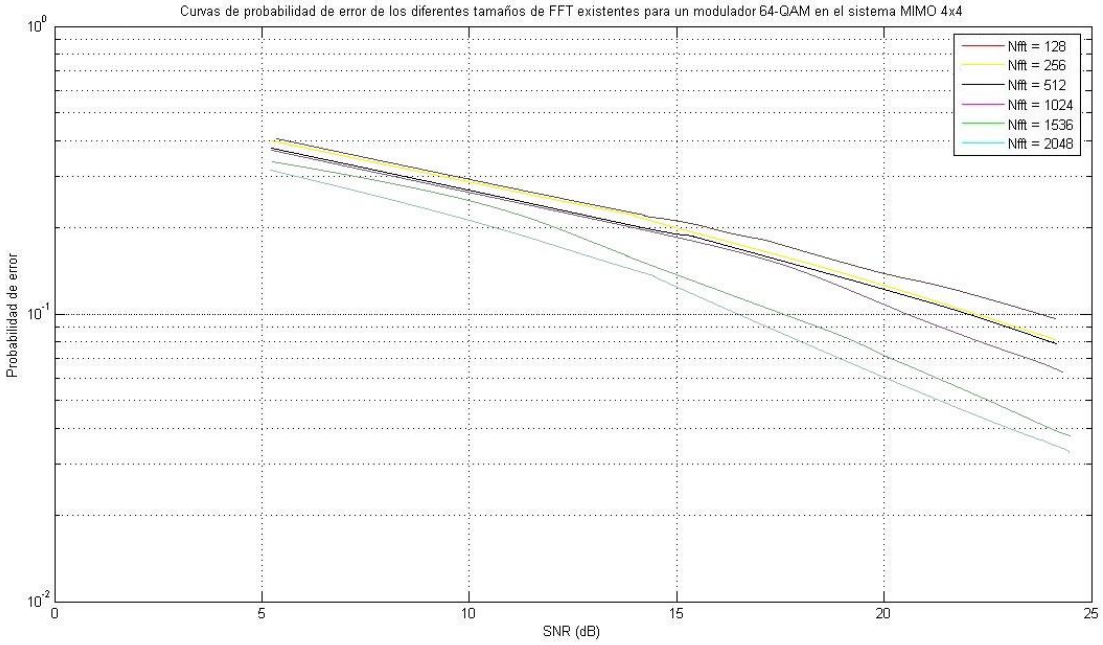

Figura 8.21 *Comparativa de las curvas de probabilidad de error en el sistema MIMO 4x4 con canal, para diferentes tamaños de FFT, con el modulador 64 QAM*

Estas gráficas reflejan como los fallos aumentan si se usa un modulador con más bits por símbolo. Continúa la reducción del aumento de errores cuando disminuye la SNR con unas curvas más próximas entre sí según aumenta el nº de bits por símbolo del modulador. Los fallos en MIMO 4x4 son menores que en el caso de MIMO 2x2.

El hecho de que las curvas a medida que aumenta el número de antenas dispongan de una forma semejante facilita la elección entre las diferentes posibilidades que ofrecen todos los sistemas, ya que la robustez frente al aumento del nivel de ruido será muy similar. Como consecuencia la selección de las diferentes tecnologías sólo estará basada en las características de cada sistema (más velocidad, mayor probabilidad de error) y no existirá ninguna tecnología más ineficiente que el resto en cuanto a su comportamiento frente al ruido. Esta similitud entre las curvas comenzó a vislumbrarse en el sistema MIMO 2x2 pero en el sistema MIMO 4x4 se hace más que evidente.

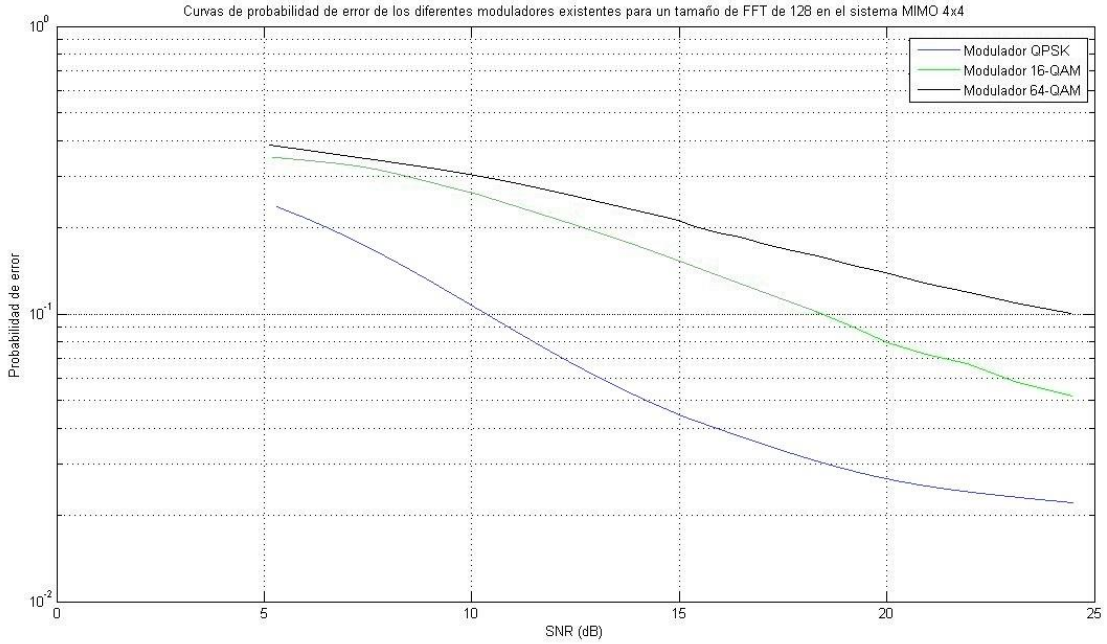

Figura 8.22 *Comparativa de las curvas de probabilidad de error en el sistema MIMO 4x4 con canal, para diferentes moduladores, con tamaño de FFT de 128*

En la gráfica anterior, se observa como los valores de probabilidad de error para los casos de los moduladores 16 QAM y 64 QAM son menores que en el sistema MIMO 2x2, esta disminución es más evidente con pocos valores de ruido. La curva del modulador QPSK es muy semejante a la del sistema con 2 antenas.

La pendiente de las curvas es menor a medida que aumenta el número de bits por símbolo del modulador, es decir, un modulador más alto implica mayor probabilidad de error pero su comportamiento frente al ruido acaba siendo más eficaz que en el caso de un modulador con menos bits por símbolo.

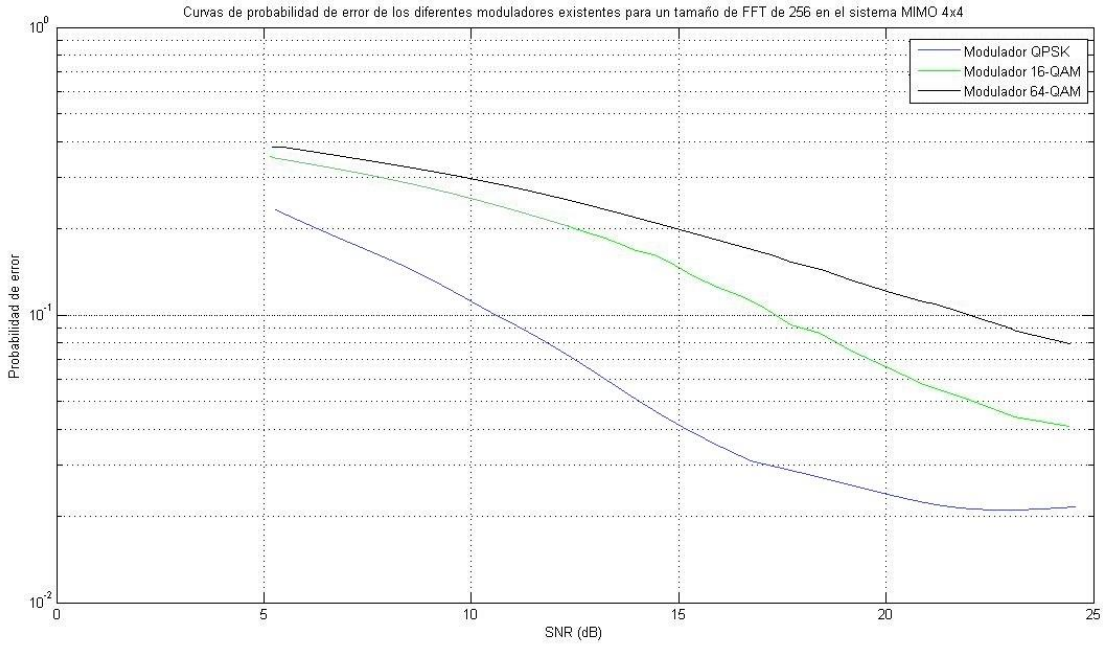

Figura 8.23 *Comparativa de las curvas de probabilidad de error en el sistema MIMO 4x4 con canal, para diferentes moduladores, con tamaño de FFT de 256*

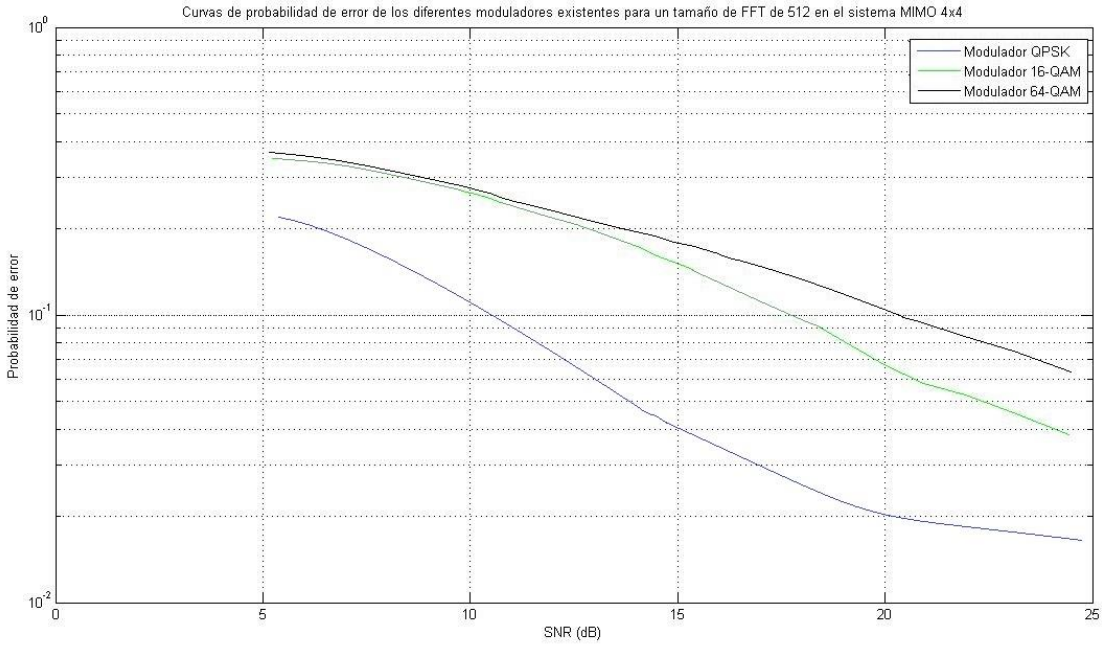

Figura 8.24 *Comparativa de las curvas de probabilidad de error en el sistema MIMO 4x4 con canal, para diferentes moduladores, con tamaño de FFT de 512*

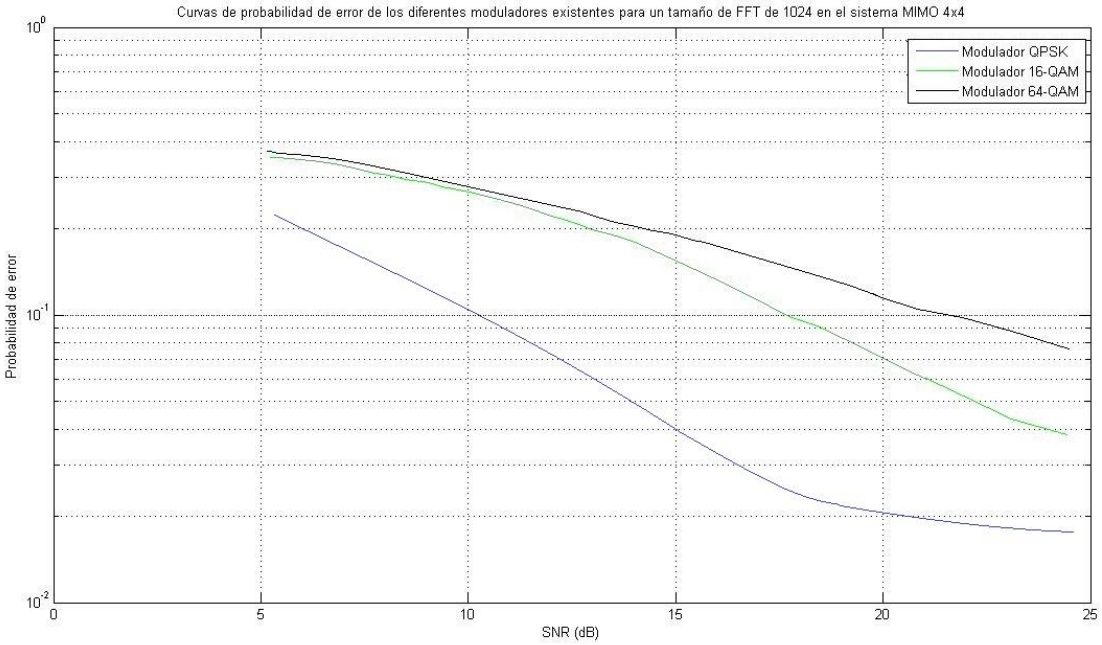

Figura 8.25 *Comparativa de las curvas de probabilidad de error en el sistema MIMO 4x4 con canal, para diferentes moduladores, con tamaño de FFT de 1024*

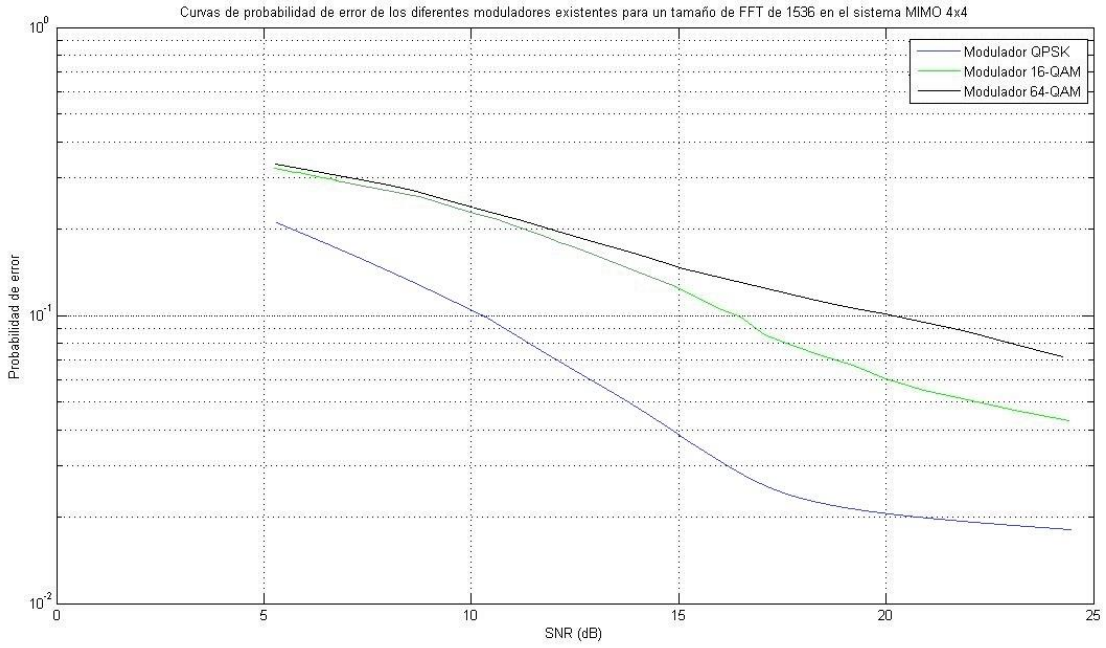

Figura 8.26 *Comparativa de las curvas de probabilidad de error en el sistema MIMO 4x4 con canal, para diferentes moduladores, con tamaño de FFT de 1536*

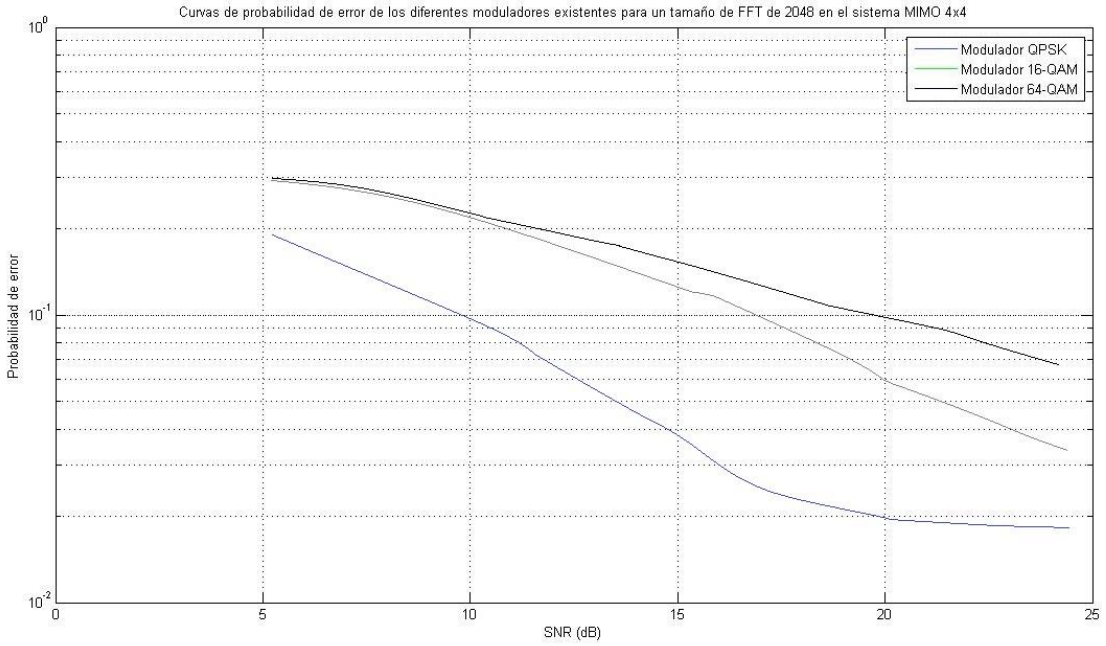

Figura 8.27 *Comparativa de las curvas de probabilidad de error en el sistema MIMO 4x4 con canal, para diferentes moduladores, con tamaño de FFT de 2048*

En todas las comparativas anteriores es evidente el paulatino descenso de la probabilidad de error a medida que aumenta el tamaño de la FFT.

Los fallos existentes en estas figuras son mayores que en el caso SISO de una única antena en el transmisor y en el receptor. Sin embargo si comparamos los resultados con las figuras del apartado 8.2 de MIMO 2x2, se puede comprobar como los resultados de probabilidad de error son siempre mayores en los casos de MIMO 2x2 cuando se usa el modulador 16 QAM o el 64 QAM. Cuando se usa un modulador QPSK sólo cuando el tamaño de FFT es alto la probabilidad de error de MIMO 4x4 es mayor.

La diferencia entre los valores de los extremos de las curvas continua acercándose, es decir, el aumento de los errores a medida que baja la relación señal a ruido continua con su proceso de disminución debido a que el sistema a medida que dispone de un mayor número de antenas es cada vez más robusto.

9. CONCLUSIONES

La combinación de las diferentes tecnologías desarrolladas en este proyecto nos lleva a la incertidumbre en cuanto a la elección entre los diferentes sistemas, cada uno de ellos con todos los posibles tamaños de FFT:

- SISO, modulador QPSK (Nfft = 128, 256, 512, 1024, 1536 ó 2048)
- SISO, modulador 16-QAM (Nfft = 128, 256, 512, 1024, 1536 ó 2048)
- SISO, modulador 64-QAM (Nfft = 128, 256, 512, 1024, 1536 ó 2048)
- MIMO 2x2, modulador QPSK (Nfft = 128, 256, 512, 1024, 1536 ó 2048)
- MIMO 2x2, modulador 16-QAM (Nfft = 128, 256, 512, 1024, 1536 ó 2048)
- MIMO 2x2, modulador 64-QAM (Nfft = 128, 256, 512, 1024, 1536 ó 2048)
- MIMO 4x4, modulador QPSK (Nfft = 128, 256, 512, 1024, 1536 ó 2048)
- MIMO 4x4, modulador 16-QAM (Nfft = 128, 256, 512, 1024, 1536 ó 2048)
- MIMO 4x4, modulador 64-QAM (Nfft = 128, 256, 512, 1024, 1536 ó 2048)

Gracias a la realización del presente proyecto y del análisis de los resultados de simulación en el apartado 8 es posible distinguir cuando es apropiado usar un sistema u otro. A continuación se citarán las diferencias más importantes entre todos los sistemas. La recopilación de estas características, repetidas a lo largo de la presente memoria, es fundamental para poder determinar cuál es el tipo de tecnología MIMO-OFDM que más se ajusta a las características de la comunicación que se pretende desarrollar. Además es muy importante tener en cuenta los valores de las curvas de probabilidad de error para conocer cuáles son los valores de la relación señal a ruido adecuados para poder usar ciertas tecnologías.

En la posterior selección de un sistema u otro se tendrán en cuenta principalmente aquellas características relacionadas con el presente proyecto, aunque también se mencionaran otros aspectos como la corrección de errores o la colocación y el coste de múltiples antenas.

El uso de más de una antena transmisora y receptora tiene como beneficio fundamental el aumento de la velocidad de transmisión. El uso del sistema MIMO 2x2 implica que la velocidad se duplica respecto al uso de la tecnología SISO. Cuando se usa MIMO con cuatro antenas transmisoras y otras cuatro receptoras la tasa de transmisión es dos veces mayor que en la tecnología MIMO 2x2 y cuatro veces mayor que usando una única antena en el transmisor y en el receptor.

Como inconveniente ante el uso de un mayor número de antenas, la probabilidad de error es mayor en los sistemas de varias antenas que en el sistema SISO. Esta desventaja tiene como compensación que la velocidad es mucho más alta. Además, el hecho de utilizar más cantidad de antenas supone una mayor robustez frente a la existencia de mayor cantidad de ruido en el canal, es decir, cuanto mayor es el número de antenas en el sistema el aumento de la probabilidad de error a medida que disminuye la relación señal a ruido es menor.

Por otro lado, en el caso MIMO 4x4 su porcentaje de errores en la mayoría de las situaciones es menor que en el caso MIMO 2x2 (como se explicó en el punto 8.3 de la memoria). Como consecuencia, en la mayoría de las situaciones en las que existen múltiples antenas prevalece el uso de la tecnología que tiene mayor número de antenas.

Un inconveniente puede surgir debido a los problemas que puede acarrear la colocación de varias antenas en un mismo terminal y el espacio que eso implica. Ese espacio tiene como consecuencia un aumento importante del tamaño de los terminales. No sólo es un problema de comodidad, ya que por razones de uso en algunas situaciones no es posible disponer de un terminal con esas medidas. Además el uso de múltiples antenas también conlleva un mayor coste económico.

Por lo tanto para el uso de múltiples antenas hay que valorar la velocidad requerida y su probabilidad de error, además de las consecuencias de introducir varias antenas en el terminal. Con respecto a la probabilidad de error, posteriormente se explicará la importancia del ruido del canal en la selección de unos sistemas u otros. Es básico conocer el máximo nivel de ruido que admite una tecnología para que funcione de forma adecuada debido a que en un canal con mucho ruido los fallos producidos pueden implicar un funcionamiento ineficiente de la tecnología.

El tamaño de FFT vendrá determinado por el ancho de banda requerido en la comunicación, como se ha explicado detalladamente en la parte teórica (en el apartado 4.4 existe una tabla con la relación entre el ancho de banda y su respectivo tamaño de FFT). Por lo tanto, el ancho de banda concertado para la comunicación determinará el tamaño de la FFT usada.

El empleo de un modulador con más bits por símbolo proporciona una mayor velocidad de transmisión, ya que en el mismo tiempo se envía una mayor cantidad de símbolos. La parte negativa es que existe una mayor cantidad de errores al encontrarse los símbolos más próximos entre sí.

Para que el uso de un modulador sea eficiente es necesario asegurarse de que se usa el modulador en unos niveles adecuados de probabilidad de error. El límite de fallos adecuado a partir del cual puede utilizarse un sistema sin problema es 10^{-1} (según las unidades usadas en las curvas de probabilidad de error del punto anterior). Hay que tener en cuenta que en una implementación real, en el receptor existe un turbo código (no desarrollado en este trabajo de fin de grado) que se encarga de la corrección de errores y permite reducir esos fallos (10^{-1}) del sistema consiguiendo una comunicación fiable.

La elección del modulador a utilizar no sólo vendrá definida por el tipo de servicio que se pretenda ofrecer sino también por la cantidad de ruido existente en la

comunicación que se pretenda establecer y su número de errores. A continuación se citarán una media aproximada de los diferentes valores de señal a ruido necesarios para garantizar la correcta funcionalidad del sistema para cada uno de los tres tipos de moduladores, esta media implica un promedio de los valores de los diferentes tamaños de FFT para los diferentes moduladores y sistemas de una o varias antenas.

En el caso del modulador QPSK, no es necesario disponer de un alto nivel de relación señal a ruido para que los errores sean menores de 10^{-1}. Su uso es posible en la mayoría de los casos, excepto por debajo de una SNR de 10-13 dB, ya que para esos valores en general, se suelen producir demasiados errores en todos los sistemas de comunicaciones independientemente de la tecnología empleada.

Cuando se pretende usar el modulador 16 QAM o el 64 QAM, es muy importante tener en cuenta el nivel de ruido del canal. El uso de cualquiera de los dos moduladores en un canal con excesivo ruido conllevaría un ineficiente funcionamiento de la tecnología desarrollada.

En el caso de la modulación 16 QAM cuando se usan múltiples antenas el valor de la relación señal a ruido debe ser mayor de 17 dB para un uso eficiente del sistema en el caso MIMO 4x4 y por encima de 20 dB en MIMO 2x2. Con la tecnología SISO si se desea que el modulador 16 QAM funcione de forma correcta es necesario disponer de un canal con una SNR mayor que 10 dB.

Si se desea usar un modulador 64 QAM con el sistema SISO se necesita una SNR mayor que 14 dB, para el sistema MIMO 2x2 debe estar por encima de 23 dB y para el caso de MIMO 4x4 unos 20 dB.

Ahora que ya han sido explicados los aspectos a tener en cuenta para la elección de un sistema u otro y las condiciones que se deben cumplir para poder seleccionarlos, se procederá a citar unos ejemplos simples de comunicaciones requeridas en la actualidad para relacionarlos con las tecnologías más adecuadas para los mismos.

En una comunicación en tiempo real el aspecto más importante a tener en cuenta es la velocidad. Es necesario conseguir la mayor velocidad posible para que la información llegue de forma instantánea al receptor. La preocupación por los errores y la protección de los mismos pasa a un segundo plano, aunque obviamente sigue siendo un aspecto a tener en cuenta.

Salvo algún problema con la implementación de las cuatro antenas dentro del terminal, lo más normal es la elección de un sistema MIMO 4x4.

El hecho de priorizar la velocidad de transmisión lleva a la elección del modulador con más bits por símbolo posible, en el caso del presente proyecto, el 64 QAM. Por lo tanto éste sería el modulador ideal para aquellas comunicaciones que requieran mucha velocidad y en las que el uso de una velocidad menor ponga en riesgo el servicio ofrecido. Sin embargo, esta elección no debe olvidar lo expuesto en párrafos anteriores sobre el nivel del ruido del canal. Es prioritario conocer si el ruido del canal no es adecuado para este modulador; en caso de ser así, habría que tratar de usar un modulador de menor número de bits por símbolo para que la probabilidad de error no supere el valor de 10^{-1} para el ruido existente en el canal.

En otras situaciones en las que la velocidad deba ser elevada pero cuya disminución no deteriore la calidad de servicio ofrecida, sería suficiente con el uso del modulador 16-QAM cuya velocidad es menor que la del 64-QAM pero que ofrece menos errores en la transmisión. Salvo que haya un problema con la colocación o con el coste de tener varias antenas, sigue siendo normal el uso del sistema MIMO 4x4 frente al MIMO 2x2 ya que la velocidad es el doble y además los errores son incluso menores.

En el caso en el que el objetivo sea establecer un envío de información más seguro sin necesitar una alta velocidad, el uso del modulador QPSK en un sistema SISO implica la más baja probabilidad de errores y un menor coste para su desarrollo, además esos fallos tan reducidos conllevan que apenas haya que preocuparse por la relación señal a ruido existente en el canal.

9. CONCLUSIONS

The combination of the different technologies developed in this project suggests a great uncertainty about the choice of the system to be used, each of them with all the possible FFT sizes.

- SISO, QPSK modulator
  (FFT size = 128, 256, 512, 1024, 1536 or 2048)

- SISO, 16-QAM modulator
  (FFT size = 128, 256, 512, 1024, 1536 or 2048)

- SISO, 64-QAM modulator
  (FFT size = 128, 256, 512, 1024, 1536 or 2048)

- MIMO 2x2, QPSK modulator
  (FFT size = 128, 256, 512, 1024, 1536 or 2048)

- MIMO 2x2, 16-QAM modulator
  (FFT size = 128, 256, 512, 1024, 1536 or 2048)

- MIMO 2x2, 64-QAM modulator
  (FFT size = 128, 256, 512, 1024, 1536 or 2048)

- MIMO 4x4, QPSK modulator
  (FFT size = 128, 256, 512, 1024, 1536 or 2048)

- MIMO 4x4, 16-QAM modulator
  (FFT size = 128, 256, 512, 1024, 1536 or 2048)

- MIMO 4x4, 64-QAM modulator
  (FFT size = 128, 256, 512, 1024, 1536 or 2048)

Due to the implementation of this project and the analysis of the simulation results in section 8 is possible to identify when it is appropriate to use one system or another. Then, the most important differences between all the systems will be explained. The collection of these characteristics is essential for determining the best MIMO-OFDM technology for the communication that is to develop. Also, it is very important to consider the values of the error probability curves to know what values of signal-to-noise ratio are correct to use certain technologies.

In the subsequent selection of one system or another, those characteristics related to this project will be considered mainly, but other aspects like error correction or the placement and cost of multiple antennas are mentioned too.

The utilization of more than one antenna in the transmitter and in the receiver enhances, as its main advantage, the transmission speed. The use of 2x2 MIMO system implies that the speed is double than in SISO technology. When MIMO is used with four transmitter antennas and four receiver antennas, the transmission rate is two times higher than in 2x2 MIMO technology and four times bigger than if a single antenna is used.

One problem referred to the use of more antennas is that there are more mistakes than in the SISO system. This disadvantage is compensated with a much higher speed. Furthermore, the fact of using more number of antennas increases robustness against the existence of more noise into the channel, that is to say, the increase of the probability of error is smaller when signal-to-noise ratio decreases in a system with more antennas.

By the other hand, in the 4x4 MIMO cases the percentage of errors in most situations is less than in 2x2 MIMO cases (this is explained in section 8.3 of this report). Consequently, in most situations with multiple antennas the use of the technology that has more number of antennas prevails.

A problem can be caused due to the placement of multiple antennas in a single terminal and space that implies. The terminals size increases because of the new space needed. It is not only a problem of comfort, because in some situations it´s not possible to have a mobile with a big size. Furthermore the use of multiple antennas also involves a high cost.

Consequently, when multiple antennas are used it is basic the assessment of the required speed and its error probability and the consequences of introducing multiple antennas at the terminal too. The importance of the channel noise in the selection of different systems will be explained later. It is essential to know the maximum noise level that supports a technology because too much noise in the channel can cause an inefficient working of the system.

The FFT size is determined by the bandwidth required in the communication, as has been explained in detail in the theoretical part (In 3.4, there is a table with the relation between bandwidth and their respective FFT size). The agreed bandwidth determines what FFT size to be used.

The use of a modulator with more bits per symbol gives a faster transmission speed because at the same time there are more symbols. The downside is that there are more mistakes because the symbols are placed closer to each other.

For an efficient use of a modulator it is necessary knowing if the modulator is used in an appropriate level of error probability. The failure threshold is 10^{-1}, with that value any systems can be used (according to the units used in error probability curves of the previous section). This value is enough because in a real implementation, there is a turbo code in the receiver (not developed in this bachelor thesis) that handles error correction, reduces these failures (10^{-1}) and makes that the system be a reliable communication.

In the case of QPSK modulator, it is not necessary to have a high signal to noise ratio to have errors less than 10^{-1}. Its use is possible in most cases, except underneath

10-13 dB of SNR, for these values generally too many mistakes are produced in all communications systems whatever of the technology employed.

When a 16-QAM or 64-QAM modulator is used, is very important to consider the noise level of the channel. The use of either modulator in a channel with an excessive noise entails an inefficient operation of the developed technology.

In the case of 16-QAM modulation the value of the signal to noise ratio must be higher than 17 dB for an efficient use of the 4x4 MIMO systems and more than 20 dB for 2x2 MIMO. It is necessary to have a channel with an SNR higher than 10 dB with SISO.

For a correct working of the 64-QAM modulator, it is required more than 14 dB in SISO system; it must be above 23 dB in 2x2 MIMO technologies and for the case of 4x4 MIMO, a minimum value of 20 dB.

Now that the aspects to consider for choosing one system or another have been explained, some simple examples of current communications will be cited, with the aim to relate them with their right technology.

The choice of the modulator is defined not only by the type of service offered but also by the amount of noise existing in the communication and the amount of mistakes. Next, an approximate average of the different values of signal to noise ratio required to ensure proper functionality of the system for each of the three types of modulators are cited , this is an average of different FFT sizes for all the types of modulators and for systems of only one antenna or more than one.

The data rate is the most transcendent aspect in a real time communication. It is necessary to achieve the fastest possible speed so that information travels instantly to the receiver. The concern about mistakes and protection are related to the background, although it continues being an aspect to consider obviously.

Except a problem with the implementation of four antennas inside the terminal, the most common it is to choose a 4x4 MIMO system.

Prioritizing the transmission rate leads to the choice of the modulator with more bits per symbol. The modulator with more bits per symbol established by the Release 8 for downlink is 64-QAM, which is the one that offers the highest possible speed because it sends 6 bits per symbol. Therefore, this modulator would be ideal for those communications that require high speed and in which the use of a lower speed endangers the service offered. However, it is essential not to forget the explanation of previous paragraphs about the channel noise. It is very important to know if the channel noise is not suitable for this modulator, in that situation, a modulator with fewer bits per symbol should be used, with the aim that error probability does not exceed the value of 10^{-1} in the same noise conditions.

It is enough to use a 16-QAM modulator, whose speed is smaller than 64-QAM rate, in other situations when the speed must be high but without decreasing the service quality offered. The 16-QAM provides a high speed with fewer mistakes than a 64-QAM modulator. Unless there is a problem with the installation or the cost of having

several antennas, it is normal to use MIMO 4x4 instead of MIMO 2x2, because its speed is the double and its mistakes are smaller in most of the situations.

The use of a QPSK modulator with the lowest possible number of mistakes is desirable when the objective is to send information safely without having to worry about the transmission rate, the use of a SISO system has the fewest percentage of mistakes and a lower cost for its development, a little amount of mistakes also implies that it's not very important to worry about the signal to noise ratio on the channel.

10. LÍNEAS DE TRABAJOS FUTUROS

Tal y como se ha explicado en el apartado 3.4, LTE-Advanced es la evolución directa de la tecnología LTE.

El sistema LTE definido en la Release 8 alcanza una velocidad de datos de bajada de 150 Mbps, registros que no eran suficientes para ser considerada como tecnología 4G. Sin embargo, LTE-Advanced sí cumple los requerimientos de la Unión Internacional de Telecomunicaciones al disponer de una velocidad de datos de bajada de 1 Gbps.

El presente trabajo está preparado para ofrecer el enlace descendente de LTE según la Release 8 de 3GPP, pero su posible adaptación futura a la tecnología LTE-Advanced sería bastante factible. Para poder conocer cómo adaptar de la forma más sencilla posible un sistema al otro es necesario tener claras las variaciones sufridas por LTE-Advanced sin olvidar que LTE es la base de la futura tecnología y que la evolución surge de una forma muy natural. Estos cambios ya han sido explicados en la presente memoria.

El aumento hasta 8 antenas en el sistema MIMO en el enlace descendente en lugar del máximo de 4 existente en la Release 8 sirve para un gran aumento de la capacidad y lograr las altas tasas de datos requeridas.

Para desarrollar esta variación basta con implementar la estructura de las nuevas rejillas de recursos para las antenas número 5, 6, 7 y 8. Estas nuevas estructuras dispondrán de unas ubicaciones de las señales de referencia propias lo que implicará que se añadan posiciones sin usar para el resto de antenas y también para estas nuevas antenas con respecto a las rejillas de recursos usadas en este proyecto.

La primera modificación sería la incorporación de nuevas funciones de las antenas añadidas para los conversores Serie-Paralelo en el transmisor. Estos conversores al igual que los existentes en la implementación actual tendrían como cometido añadir las posiciones de las señales pilotos y de las posiciones no usadas, manteniendo el espacio de guarda definido por el estándar en función del tamaño de FFT que se desee usar en la transmisión.

El siguiente cambio se produciría en el receptor. Sería necesario desarrollar un ecualizador para cada uno de las nuevas antenas. Estos ecualizadores realizarían la estimación del canal para cada una de las antenas teniendo en cuenta sus determinadas posiciones piloto en el estándar. A partir de todas las matrices obtenidas (cada ecualizador en cada uno de los flujos de información) se realizaría una pseudoinversa de todas ellas para corregir las variaciones del canal y conseguir que la información que llega al ecualizador sea lo más parecida posible a la que fue enviada desde el transmisor.

Más tarde habría que modificar el conversor Paralelo-Serie del receptor, aunque puede seguirse usando un único conversor para todos los flujos de información destinados a cada uno de los receptores, al igual que sucede en el sistema MIMO 2x2 y

MIMO 4x4. Sería necesario adaptar este conversor a las características de un sistema de 8 antenas, ya que el número de posiciones pilotos y por lo tanto de las posiciones no usadas ha aumentado.

Los moduladores usados en el sistema serían los mismos que están desarrollados en el presente trabajo, es decir, QPSK, 16-QAM y 64-QAM para el enlace descendente. En un principio no se utilizarían moduladores con un mayor número de bits por símbolo.

La realización de las medidas correspondientes a los resultados de simulación y validación tendría el mismo proceso que el realizado en los actuales sistemas (SISO, MIMO 2x2 y MIMO 4x4), ya explicado en el apartado 6.2.4, sólo que para calcular los errores de las curvas para diferentes valores de ruido habría que realizar el proceso con un promedio de los errores de cada una de las 8 antenas. Después la forma de organizar las gráficas y sus comparativas sería la misma, ya que es la forma más eficiente de poder explicar cómo varía el sistema con el uso de los diferentes tipos de moduladores y con la utilización de los tamaños de FFT posibles y así comparar las variaciones surgidas con respecto a la tecnología LTE del presente proyecto.

11. PLANIFICACIÓN

El presente trabajo fin de grado comenzó a elaborarse en Febrero de 2013. Las primeras tareas consistieron en una documentación sobre los aspectos más relevantes de las tecnologías OFDM y MIMO. Esta primera toma de contacto sirvió para entender de forma general estas dos tecnologías y para seleccionar parte de la información a usar durante el resto del proyecto.

La siguiente fase consistió en seleccionar con más detalle los aspectos más útiles de OFDM para realizar en Matlab el sistema SISO. Para ello fue fundamental entender las funciones de todos los bloques de OFDM y seleccionar del estándar la estructura de la rejilla de recursos. Después de este ejercicio de comprensión se produjo la implementación del sistema. En un primer momento se procedió a trabajar con un canal simple con dos multitrayectos sin ruido.

Antes de la realización del ecualizador tal y como está definido en el estándar se realizaron diversos prototipos de ecualizadores, sin basarse en ninguno teórico ya realizado, con el objetivo de desarrollar formas originales de usar las señales de referencia de la matriz y tratar de ir reduciendo los errores del sistema. Además se elaboró una alternativa de diseño al modelo teórico variando las frecuencias útiles tal y como se explicó en el apartado 6.

Antes de empezar con los otros sistemas fue necesario ampliar los conocimientos de MIMO y conocer las estructuras de sus rejillas de recursos para cada una de las antenas usadas. Se comenzó usando la base desarrollada en el sistema SISO usando los bloques OFDM en cada una de las antenas con su respectiva matriz de recursos. En un primer momento se trabajó con los flujos de información de cada una de las antenas separados hasta comprobar que el transporte de todos ellos hacia el receptor de forma independiente era correcto. A partir de la constatación del comportamiento deseado se realizó la mezcla de las diferentes señales procedentes de las antenas en el canal, tal y como sucede en la realidad. Debido a esta mezcla de información fue básico comprender el funcionamiento del receptor Zero-Forcing y desarrollarlo en el ecualizador de los sistemas MIMO.

Tras la correcta elaboración de todas las tecnologías el siguiente paso fue simular y usar las respectivas matrices del canal SCM para cada uno de los sistemas. Continuando el proceso de convertir el canal utilizado en un canal realista se desarrolló la forma de normalizar cualquier canal usado y se añadió ruido, la variación de la amplitud del ruido determina la relación señal a ruido usada por la comunicación.

Estando ya listos todos los sistemas con unos canales realistas se procedió a obtener las gráficas de validación y su simulación. A partir de ellas se realizaron sus pertinentes comentarios. Parte de este análisis constituye la base para el apartado de las conclusiones de la memoria.

Por último se procedió a la realización de esta memoria que engloba todos los procesos explicados en este apartado. Para su preparación aparte de la información usada hasta el momento también fue necesaria la recopilación de más aspectos teóricos para contextualizar la tecnología LTE y su evolución dentro de las comunicaciones móviles.

12. PRESUPUESTO

El presupuesto está formado por:

- **Coste de equipo**

Para la realización del proyecto se ha optado por un ordenador portátil de gama media con las siguientes características:

- Fabricante: Acer
- Modelo: Aspire E1-531
- Procesador: Intel Core i5 3230 (2.6 GHz)
- Memoria RAM: 6 GB

Precio: 499,00 €

También es necesario disponer de una licencia Matlab, la versión para estudiantes tiene un precio de 69 €. [16]

ORDENADOR PORTÁTIL	499,00 €
Licencia Matlab	69,00 €
TOTAL COSTE DE EQUIPO	**568,00 €**

- **Coste de personal**

Esta parte del presupuesto reflejará el trabajo realizado por el ingeniero encargado de realizar el trabajo fin de grado y el tutor o jefe de proyecto encargado de supervisar el mismo.

Para ello es necesario indicar las horas trabajadas y el salario por hora para cada uno.

El salario está basado en el coste de la contratación laboral en la Universidad Carlos III de Madrid en el año 2012.

CARGO	TIEMPO	SALARIO	TOTAL
Ingeniero	350 horas	24,70 € / hora	9.695 €
Jefe de proyecto	32 horas	37,25 € / hora	1.192 €
TOTAL COSTE DE PERSONAL			**10.887 €**

- **Coste total** (IVA incluido)

COSTE TOTAL	**11.455 €**

ANEXO

Código realizado

CÓDIGO DEL SISTEMA SISO

Archivos necesarios:

- anadirPrefijoCiclico.m
- conversorPSrx2.m
- conversorPStx.m
- conversorSPrx.m
- conversorSPtxMaximosPilotos.m
- conversorSPtxPilotosEstandar.m
- Demodulador16QAM.m
- Demodulador64QAM.m
- DemoduladorQPSK.m
- ecualizadorMatrizEnteraAntena1.m
- ElegirDemodulador.m
- ElegirModulador.m
- eliminarPrefijoCiclico.m
- Fftrx.m
- Iffttx.m
- Modulador16QAM.m
- Modulador64QAM.m
- ModuladorQPSK.m
- Siso.m

A continuación se muestra el script del código elaborado para el sistema SISO (Siso.m) con una antena transmisora y con otra antena receptora.
Posteriormente se muestran el resto de funciones.

```
--------------------------------------------------------------------------------

clc
clear all
close all

% Sistema OFDMA

% Bits de entrada que enviamos desde el transmisor al receptor
```

```
bitsEntrada = round(rand(1,16812*1));

% 1 - TRANSMISOR

% 1.1 - Modulador: Convertimos los bits de entrada en símbolos
% Elegimos la constelación que queremos mediante la variable tipoConstelacion:
% 1 - Constelación QPSK, 2 - Constelación 16QAM y 3 - Constelación 64QAM
tipoConstelacion = 3;
salidaModulador = ElegirModulador(bitsEntrada,tipoConstelacion);
PotenciaSenal = var(salidaModulador); %Medimos la potencia de la señal

% 1.2 - Conversor Serie Paralelo
Nfft = 512; % Definimos el tamaño de la FFT
P = 1; % Valor de la señal piloto a introducir
% Usamos las frecuencias útiles definidas en el estándar
[GP,Npilotos,salidaSPtx] = conversorSPtxPilotosEstandar(salidaModulador,Nfft,P);
% Usamos las máximas frecuencias útiles posibles
%[GP,Npilotos,salidaSPtx] = conversorSPtxMaximosPilotos(salidaModulador,Nfft,P);

% 1.3 - IFFT: Pasamos del dominio de la frecuencia al dominio del tiempo
salidaifft = iffttx(salidaSPtx,Nfft);

% 1.4 - Añadir prefijo cíclico
% La adición del prefijo cíclico consiste en copiar la parte final del
% símbolo al principio del mismo, es decir, el final de la columna del símbolo
% correspondiente, en el principio de la misma
tamPC = Nfft/8; % Tamaño del prefijo cíclico
[filas,columnas] = size(salidaifft);
salidaPCtx = zeros(filas+tamPC,columnas);
for n=1:columnas
   salidaPCtx(:,n) = anadirPrefijoCiclico(salidaifft(:,n),tamPC);
end

% 1.5 - Conversor Paralelo Serie
salidaPStx = conversorPStx(salidaPCtx);

% 2 - CANAL: Definimos el canal existente entre el transmisor y el receptor
%canal = 1;
canal=[0.909505559686059+0.0570811522831248i 0.0456760830213736+0.111242422458266i
-0.247602286506910-0.009283304416878O5i 0.0610869817003726-0.0342405687499177i -
0.008029148864771 59+0.0224937836985498i 0.153191551329004+0.1308318124832 16i];
salidaCanal = filter(canal, 1, salidaPStx);
PotenciaSenal1 = var(salidaCanal);
norm = PotenciaSenal/PotenciaSenal1;
canal = canal.*sqrt(norm); %Normalizamos el canal
salidaCanal = filter(canal, 1, salidaPStx);
PotenciaSenal1 = var(salidaCanal); %Potencia de la señal
% Si se desea se añade ruido
ni1 = 0;
ni =
(ni1)*(1/sqrt(2))*(randn(1,length(salidaPStx))+1i*randn(1,length(salidaPStx)));
PotenciaRuido = var(ni);
salidaCanalConRuido = salidaCanal+ni;
% Nivel Señal a Ruido
SNRdB = 10*log10(PotenciaSenal/PotenciaRuido);

% 3 - RECEPTOR

% 3.1 - Conversor Serie Paralelo
salidaSPrx = conversorSPrx(salidaCanalConRuido,Nfft,tamPC);

% 3.2 - Eliminar Prefijo Cíclico
% Eliminamos el comienzo de cada una de los columnas
% El tamaño del prefijo cíclico está definido anteriormente en tamPC
[filas,columnas] = size(salidaSPrx);
salidaPCrx = zeros(filas-tamPC,columnas);
```

```
for n=1:columnas
   salidaPCrx(:,n) = eliminarPrefijoCiclico(salidaSPrx(:,n),tamPC);
end

% 3.1 - FFT
% Pasamos del dominio del tiempo al dominio frecuencial
salidafft = fftrx(salidaPCrx,Nfft);

% 3.2 - Ecualizador: Revierte el impacto del canal
% Obtenemos la matriz H que será la encargada de indicarnos la variación de
% amplitud y fase que ha sufrido cada uno de los elementos de los
% diferentes símbolos, se multiplica cada elemento de la matriz H (un número
complejo
% basado en la estimación del canal) por su elemento correspondiente en la
% matriz que llega al ecualizador
H = ecualizadorMatrizEnteraAntena1(salidafft,P,GP);
salidaEcualizador = P*salidafft./(H);

% Conversor Paralelo Serie
salidaPSrx = conversorPSrx2(salidaEcualizador,GP,Npilotos);
salidaPSrx = salidaPSrx(1:length(salidaModulador));

% Demodulador
% Convierte los símbolos en una cadena de bits
salidaDemodulador =  ElegirDemodulador(salidaPSrx,tipoConstelacion);
% La salida se corresponderá con una cadena de bits del mismo tamaño que la
% enviada a través del transmisor
salidaDemodulador = salidaDemodulador(1:length(bitsEntrada));

% Calculamos el número de fallos y el porcentaje de los mismos respecto a
% los bits de entrada y los bits de salida obtenidos
fallos = 0;
for i=1:length(bitsEntrada)
    b = (abs(salidaDemodulador(i)-bitsEntrada(i)));
    fallos = fallos+b;
end
PorcentajeFallos = (fallos*100)/length(bitsEntrada);

% Figura de la estimación del canal
figure(1)
plot(abs(fft(canal,Nfft)))
grid on
hold on
plot(abs(H(:,1)),'r')

--------------------------------------------------------------------------------

function [salidaModulador] = ElegirModulador(bitsEntrada,tipoConstelacion)

% Elegimos la constelación que queremos:
% 1 - Constelación QPSK
% 2 - Constelación 16QAM
% 3 - Constelación 64QAM

switch(tipoConstelacion)
    case 1
        salidaModulador = ModuladorQPSK(bitsEntrada);
    case 2
        salidaModulador = Modulador16QAM(bitsEntrada);
    case 3
        salidaModulador = Modulador64QAM(bitsEntrada);
end
end

--------------------------------------------------------------------------------
```

```
function [salidaModulador]= ModuladorQPSK(bitstx)

N=length(bitstx);
salidaModulador = zeros(1,N/2);

aux = 1;
for n=1:2:N
    if (bitstx(n)==0)
        if (bitstx(n+1)==0)
            salidaModulador(1,aux) = (1+1i)/sqrt(2);
        else
            salidaModulador(1,aux)  =(1-1i)/sqrt(2);
        end
    else
        if (bitstx(n+1)==0)
            salidaModulador(1,aux) = (-1+1i)/sqrt(2);
        else
            salidaModulador(1,aux)=(-1-1i)/sqrt(2);
        end
    end

    aux = aux+1;
end
end

--------------------------------------------------------------------------------

function [salidaModulador]= Moduladror16QAM(bitstx)

N=length(bitstx);
salidaModulador = zeros(1,N/4);

aux = 1;

for n = 1:4:N
    if (bitstx(n)==0)
        if(bitstx(n+1)==0)
            if(bitstx(n+2)==0)
                if(bitstx(n+3)==0)
                    salidaModulador(1,aux)=(1+1i)/sqrt(10); %0000
                else
                    salidaModulador(1,aux)=(1+3*1i)/sqrt(10); %0001
                end
            else
                if(bitstx(n+3)==0)
                    salidaModulador(1,aux)=(3+1i)/sqrt(10); %0010
                else
                    salidaModulador(1,aux)=(3+3*1i)/sqrt(10); %0011
                end
            end
        else
            if(bitstx(n+2)==0)
                if(bitstx(n+3)==0)
                    salidaModulador(1,aux)=(1-3*1i)/sqrt(10); %0100
                else
                    salidaModulador(1,aux)=(1-1i)/sqrt(10); %0101
                end
            else
                if(bitstx(n+3)==0)
                    salidaModulador(1,aux)=(3-3*1i)/sqrt(10); %0110
                else
                    salidaModulador(1,aux)=(3-1i)/sqrt(10); %0111
                end
            end
        end
    else
```

```
            if(bitstx(n+1)==0)
             if(bitstx(n+2)==0)
                 if(bitstx(n+3)==0)
                     salidaModulador(1,aux)=(-3+1i)/sqrt(10); %1000

                 else
                     salidaModulador(1,aux)=(-3+3*1i)/sqrt(10); %1001
                 end
             else
                 if(bitstx(n+3)==0)
                     salidaModulador(1,aux)=(-1+1i)/sqrt(10); %1010;
                 else
                     salidaModulador(1,aux)=(-1+3*1i)/sqrt(10); %1011
                 end
            end
        else
             if(bitstx(n+2)==0)
                 if(bitstx(n+3)==0)
                     salidaModulador(1,aux)=(-3-3*1i)/sqrt(10); %1100
                 else
                     salidaModulador(1,aux)=(-3-1i)/sqrt(10); %1101
                 end
             else
                 if(bitstx(n+3)==0)
                     salidaModulador(1,aux)=(-1-3*1i)/sqrt(10); %1110
                 else
                     salidaModulador(1,aux)=(-1-1i)/sqrt(10); %1111
                 end
             end
            end
    end
    aux = aux+1;
end
end

-------------------------------------------------------------------------------

function [salidaModulador]= Modulador64QAM(bitstx)

N=length(bitstx);
salidaModulador = zeros(1,N/6);

aux = 1;

for n = 1:6:N
    if (bitstx(n)==0)
        if(bitstx(n+1)==0)
            if(bitstx(n+2)==0)
                if(bitstx(n+3)==0)
                    if(bitstx(n+4)==0)
                        if(bitstx(n+5)==0)
                           salidaModulador(1,aux)=(3+3i)/sqrt(42); %000000
                        else
                            salidaModulador(1,aux)=(3+1i)/sqrt(42); %000001
                        end
                    else
                        if(bitstx(n+5)==0)
                           salidaModulador(1,aux)=(3+5i)/sqrt(42); %000010
                        else
                            salidaModulador(1,aux)=(3+7i)/sqrt(42); %000011
                        end
                    end
                else
                    if(bitstx(n+4)==0)
                        if(bitstx(n+5)==0)
                           salidaModulador(1,aux)=(3-3i)/sqrt(42); %000100
```

```
                else
                    salidaModulador(1,aux)=(3-1i)/sqrt(42); %000101
                end
            else
                if(bitstx(n+5)==0)
                   salidaModulador(1,aux)=(3-5i)/sqrt(42); %000110
                else
                    salidaModulador(1,aux)=(3-7i)/sqrt(42); %000111
                end
            end
        end
    else
        if(bitstx(n+3)==0)
            if(bitstx(n+4)==0)
                if(bitstx(n+5)==0)
                   salidaModulador(1,aux)=(1+3i)/sqrt(42); %001000
                else
                    salidaModulador(1,aux)=(1+1i)/sqrt(42); %001001
                end
            else
                if(bitstx(n+5)==0)
                   salidaModulador(1,aux)=(1+5i)/sqrt(42); %001010
                else
                    salidaModulador(1,aux)=(1+7i)/sqrt(42); %001011
                end
            end
        else
            if(bitstx(n+4)==0)
                if(bitstx(n+5)==0)
                   salidaModulador(1,aux)=(1-3i)/sqrt(42); %001100
                else
                    salidaModulador(1,aux)=(1-1i)/sqrt(42); %001101
                end
            else
                if(bitstx(n+5)==0)
                   salidaModulador(1,aux)=(1-5i)/sqrt(42); %001110
                else
                    salidaModulador(1,aux)=(1-7i)/sqrt(42); %001111
                end
            end
        end
    end
else
    if(bitstx(n+2)==0)
        if(bitstx(n+3)==0)
            if(bitstx(n+4)==0)
                if(bitstx(n+5)==0)
                   salidaModulador(1,aux)=(5+3i)/sqrt(42); %010000
                else
                    salidaModulador(1,aux)=(5+1i)/sqrt(42); %010001
                end
            else
                if(bitstx(n+5)==0)
                   salidaModulador(1,aux)=(5+5i)/sqrt(42); %010010
                else
                    salidaModulador(1,aux)=(5+7i)/sqrt(42); %010011
                end
            end
        else
            if(bitstx(n+4)==0)
                if(bitstx(n+5)==0)
                   salidaModulador(1,aux)=(5-3i)/sqrt(42); %010100
                else
                    salidaModulador(1,aux)=(5-1i)/sqrt(42); %010101
                end
            else
```

```
                    if(bitstx(n+5)==0)
                       salidaModulador(1,aux)=(5-5i)/sqrt(42); %010110
                    else
                        salidaModulador(1,aux)=(5-7i)/sqrt(42); %010111
                    end
                end
            end
        else
            if(bitstx(n+3)==0)
                if(bitstx(n+4)==0)
                    if(bitstx(n+5)==0)
                       salidaModulador(1,aux)=(7+3i)/sqrt(42); %011000
                    else
                        salidaModulador(1,aux)=(7+1i)/sqrt(42); %011001
                    end
                else
                    if(bitstx(n+5)==0)
                       salidaModulador(1,aux)=(7+5i)/sqrt(42); %011010
                    else
                        salidaModulador(1,aux)=(7+7i)/sqrt(42); %011011
                    end
                end
            else
                if(bitstx(n+4)==0)
                    if(bitstx(n+5)==0)
                       salidaModulador(1,aux)=(7-3i)/sqrt(42); %011100
                    else
                        salidaModulador(1,aux)=(7-1i)/sqrt(42); %011101
                    end
                else
                    if(bitstx(n+5)==0)
                       salidaModulador(1,aux)=(7-5i)/sqrt(42); %011110
                    else
                        salidaModulador(1,aux)=(7-7i)/sqrt(42); %011111
                    end
                end
            end
        end
    end
else
    if(bitstx(n+1)==0)
        if(bitstx(n+2)==0)
            if(bitstx(n+3)==0)
                if(bitstx(n+4)==0)
                    if(bitstx(n+5)==0)
                       salidaModulador(1,aux)=(-3+3i)/sqrt(42); %100000
                    else
                        salidaModulador(1,aux)=(-3+1i)/sqrt(42); %100001
                    end
                else
                    if(bitstx(n+5)==0)
                       salidaModulador(1,aux)=(-3+5i)/sqrt(42); %100010
                    else
                        salidaModulador(1,aux)=(-3+7i)/sqrt(42); %100011
                    end
                end
            else
                if(bitstx(n+4)==0)
                    if(bitstx(n+5)==0)
                       salidaModulador(1,aux)=(-3-3i)/sqrt(42); %100100
                    else
                        salidaModulador(1,aux)=(-3-1i)/sqrt(42); %100101
                    end
                else
                    if(bitstx(n+5)==0)
                       salidaModulador(1,aux)=(-3-5i)/sqrt(42); %100110
```

```
                else
                    salidaModulador(1,aux)=(-3-7i)/sqrt(42); %100111
                end
            end
        end
    else
        if(bitstx(n+3)==0)
            if(bitstx(n+4)==0)
                if(bitstx(n+5)==0)
                    salidaModulador(1,aux)=(-1+3i)/sqrt(42); %101000
                else
                    salidaModulador(1,aux)=(-1+1i)/sqrt(42); %101001
                end
            else
                if(bitstx(n+5)==0)
                    salidaModulador(1,aux)=(-1+5i)/sqrt(42); %101010
                else
                    salidaModulador(1,aux)=(-1+7i)/sqrt(42); %101011
                end
            end
        else
            if(bitstx(n+4)==0)
                if(bitstx(n+5)==0)
                    salidaModulador(1,aux)=(-1-3i)/sqrt(42); %101100
                else
                    salidaModulador(1,aux)=(-1-1i)/sqrt(42); %101101
                end
            else
                if(bitstx(n+5)==0)
                    salidaModulador(1,aux)=(-1-5i)/sqrt(42); %101110
                else
                    salidaModulador(1,aux)=(-1-7i)/sqrt(42); %101111
                end
            end
        end
    end
else
    if(bitstx(n+2)==0)
        if(bitstx(n+3)==0)
            if(bitstx(n+4)==0)
                if(bitstx(n+5)==0)
                    salidaModulador(1,aux)=(-5+3i)/sqrt(42); %110000
                else
                    salidaModulador(1,aux)=(-5+1i)/sqrt(42); %110001
                end
            else
                if(bitstx(n+5)==0)
                    salidaModulador(1,aux)=(-5+5i)/sqrt(42); %110010
                else
                    salidaModulador(1,aux)=(-5+7i)/sqrt(42); %110011
                end
            end
        else
            if(bitstx(n+4)==0)
                if(bitstx(n+5)==0)
                    salidaModulador(1,aux)=(-5-3i)/sqrt(42); %110100
                else
                    salidaModulador(1,aux)=(-5-1i)/sqrt(42); %110101
                end
            else
                if(bitstx(n+5)==0)
                    salidaModulador(1,aux)=(-5-5i)/sqrt(42); %110110
                else
                    salidaModulador(1,aux)=(-5-7i)/sqrt(42); %110111
                end
            end
```

```
                end
            else
                if(bitstx(n+3)==0)
                    if(bitstx(n+4)==0)
                        if(bitstx(n+5)==0)
                           salidaModulador(1,aux)=(-7+3i)/sqrt(42); %111000
                        else
                            salidaModulador(1,aux)=(-7+1i)/sqrt(42); %111001
                        end
                    else
                        if(bitstx(n+5)==0)
                           salidaModulador(1,aux)=(-7+5i)/sqrt(42); %111010
                        else
                            salidaModulador(1,aux)=(-7+7i)/sqrt(42); %111011
                        end
                    end
                else
                    if(bitstx(n+4)==0)
                        if(bitstx(n+5)==0)
                           salidaModulador(1,aux)=(-7-3i)/sqrt(42); %111100
                        else
                            salidaModulador(1,aux)=(-7-1i)/sqrt(42); %111101
                        end
                    else
                        if(bitstx(n+5)==0)
                           salidaModulador(1,aux)=(-7-5i)/sqrt(42); %111110
                        else
                            salidaModulador(1,aux)=(-7-7i)/sqrt(42); %111111
                        end
                    end
                end
            end
        end
    end


    aux = aux+1;
end
end
```

```
function [GP,Npilotos,salidaSPtx] = conversorSPtxPilotosEstandar
(salidaModulador,Nfft,P)

% Convierte los símbolos que salen en serie del modulador en una matriz,
% cuyas filas se corresponden con las diferentes frecuencias portadoras y
% cuyas columnas se corresponden con los slots de tiempo.
% El número de filas vendrá definido por el valor de Nfft y existirán unas
% frecuencias de guarda.
% Se insertaran las señales piloto (de valor P) que servirán para la
% estimación del canal en el ecualizador
% Cada símbolo es una columna completa definida en un slot de tiempo
% concreto

n = length(salidaModulador);
Npilotos = ceil(8*n/(12*14));
filas = Nfft;
switch (Nfft)
    case 128
        GP = (Nfft-72)/2;
    case 256
        GP = (Nfft-180)/2;
    case 512
        GP = (Nfft-300)/2;
```

```
    case 1024
        GP = (Nfft-600)/2;
    case 1536
        GP = (Nfft-900)/2;
    case 2048
        GP = (Nfft-1200)/2;
end

columnas = ceil((n+Npilotos)/(filas-2*GP))+1;
salidaSPtx = zeros(filas,columnas);

% Definimos la ubicación de las señales piloto
a = zeros(1,ceil(filas/6));
k = zeros(1,ceil(filas/6));
b = zeros(1,ceil(columnas/7));
l = zeros(1,ceil(columnas/7));
p = 1;

for i=(GP+0):6:(filas+0-GP)
    for j=0:7:columnas

        aa = 3+i;
        a(1,p) = aa;

        c = 6+i;
        k(1,p) = c;

        bb = 5+j;
        b(1,p) = bb;

        d = 1+j;
        l(1,p) = d;

        p = p+1;

    end
end

% Se rellena la matriz con el tráfico y con las señales piloto de referencia

q = 1;
for p=1:columnas
    for o=(GP+1):(filas-GP)
        % Recorremos la matriz
        % Si llegamos a una posición de la matriz correspondiente a una
        % señal piloto rellenamos la posición con el valor P, en caso
        % contrario introducimos el símbolo de tráfico del modulador que
        % corresponda

        for m=1:length(k)
            if ((k(m)==o && l(m)==p))
                salidaSPtx(o,p)=P;
                q = q-1;
                break;
            elseif (a(m)==o && b(m)==p)
                salidaSPtx(o,p)=P;
                q = q-1;
                break;
            else
                salidaSPtx(o,p)=salidaModulador(q);
            end
        end
        q = q+1;
        if (q==(length(salidaModulador)+1))
            break;
        end
```

```
    end
    if (q==(length(salidaModulador)+1))
        break;
    end
end

% Rellenamos con 0´s hasta que el número de columnas totales sea múltiplo
% de 7
columnas0 = 7*ceil(columnas/7) - columnas;
salidaSPtx = [salidaSPtx zeros(filas,columnas0)];
[filas1,columnas1] = size(salidaSPtx);

a = zeros(1,ceil(filas1/6));
k = zeros(1,ceil(filas1/6));
b = zeros(1,ceil(columnas/7));
l = zeros(1,ceil(columnas/7));
r = 1;

for m=(GP+0):6:(filas1+0-GP)
    for n=0:7:columnas1

        aa = 3+m;
        a(1,r) = aa;

        c = 6+m;
        k(1,r) = c;

        bb = 5+n;
        b(1,r) = bb;

        d = 1+n;
        l(1,r) = d;

        r = r+1;

    end
end

for p=1:columnas1
    for o=(GP+1):(filas-GP)

        for m=1:length(k)
            if ((k(m)==o && l(m)==p))
                salidaSPtx(o,p)=P;
                break;
            elseif (a(m)==o && b(m)==p)
                salidaSPtx(o,p)=P;
                break;
            else
                salidaSPtx(o,p)=salidaSPtx(o,p);
            end
        end
    end
end
end
```

```
function [salidaifft] = iffttx (salidaSP,Nfft)

% Se pasa del dominio frecuencial al temporal
```

```
[filas,columnas] = size(salidaSP);
salidaifft = zeros(filas,columnas);
for n=1:columnas
    salidaifft(:,n) = ifft(salidaSP(:,n),Nfft);
end
end

-------------------------------------------------------------------------------

function [salidaPC] = anadirPrefijoCiclico(salidaifft,tamPC)
N = length(salidaifft);
salidaPC = [salidaifft((N-tamPC+1):N); salidaifft];
end

-------------------------------------------------------------------------------

function [salidaPStx] = conversorPStx (salidaPCtx)
[filas,columnas] = size(salidaPCtx);
salidaPStx = zeros(1,filas*columnas);
for n=1:columnas
    a = salidaPCtx(:,n);
    b = conj(a');
    salidaPStx = [salidaPStx b];
end
salidaPStx = salidaPStx((filas*columnas+1):end);
end

-------------------------------------------------------------------------------

function [salidaSPrx] = conversorSPrx (salidaCanal, Nfft,tamPC)
% Convierte el vector lineal que llega desde el canal en una matriz
n = length(salidaCanal);
% Calculamos el tamaño de la matriz en función de lo realizado en el
% conversor S/P del tx y en la adición del prefijo cíclico
filas = Nfft+tamPC;
columnas = ceil(n/filas);
salidaSPrx = zeros(filas,columnas);
col = 1;
for i=0:filas:(n-filas)
    a = salidaCanal(1,(1+i):(filas+i));
    b = conj(a');
    salidaSPrx(:,col) = b;
    col = col+1;
end
end

-------------------------------------------------------------------------------

function [salidaPC] = eliminarPrefijoCiclico(salidaSP, tamPC)
N = length(salidaSP);
salidaPC = salidaSP((tamPC+1):N);
end

-------------------------------------------------------------------------------

function [salidafft] = fftrx (salidaPCrx,Nfft)

% Se pasa del dominio del tiempo al dominio frecuencial

[filas,columnas] = size(salidaPCrx);
salidafft = zeros(filas,columnas);
for n=1:columnas
    salidafft(:,n) = fft(salidaPCrx(:,n),Nfft);
end
end
```

```
--------------------------------------------------------------------------------

function [H] = ecualizadorMatrizEnteraAntena1 (salidafft,P,GP)

[filas,columnas] = size(salidafft);
h = zeros(filas-2*GP,columnas);

% Número pilotos por columna
NP = floor(2*(filas-2*GP)/12);

for j=0:7:(columnas-7)

        % 1ª columna
        a = zeros(NP,1);
        posicion1 = 6;
        for i=1:NP
            if ((GP+posicion1)<(filas-GP+1))
                a(i,1)= salidafft(GP+posicion1,j+1);
                posicion1 = posicion1+6;
            else
                break;
            end
        end
        aifft = ifft(a);
        aifft = [aifft ;zeros((filas-NP-2*GP),1)];
        afft = fft(aifft);
        h(:,j+1) = afft;

        %5ª columna
        b = zeros(NP,1);
        posicion5 = 3;
        for k=1:NP
            if ((GP+posicion5)<(filas-GP+1))
                b(k,1)= salidafft(GP+posicion5,j+5);
                posicion5 = posicion5+6;
            else
                break;
            end
        end
        bifft = ifft(b);
        bifft = [bifft ;zeros((filas-NP-2*GP),1)];
        bfft = fft(bifft);
        h(:,j+5) = bfft;
end

        % Estimación en la dimensión temporal

        % INTERPOLACIONES

        % Interpolación 1ª y 5ª columna

        for n=1:(filas-2*GP)
            for j=0:7:(columnas-7)

            % Interpolación 1ª y 5ª columna

            a1 = h(n,1+j);
            b1 = h(n,5+j);
            x1 = [1 5]; %pos. conocidas
            y1real = [real(a1) real(b1)]; %valores de dichas posiciones
            z1real = interp1(x1,y1real,[2 3 4]);
            y1imag = [imag(a1) imag(b1)];
            z1imag = interp1(x1,y1imag,[2 3 4]);
            z21 = z1real(1) +1i*z1imag(1);
            z31 = z1real(2) +1i*z1imag(2);
```

```
            z41 = z1real(3) +1i*z1imag(3);
            h(n,2+j) = z21;
            h(n,3+j) = z31;
            h(n,4+j) = z41;

            % Interpolación 5ª y 8ª columna

            if (j==(columnas-7))
                break;
            else
            a2 = h(n,5+j);
            b2 = h(n,8+j);
            x2 = [1 4];%pos. conocidas
            y2real = [real(a2) real(b2)];%valores de dichas posiciones
            z2real = interp1(x2,y2real,[2 3]);
            y2imag = [imag(a2) imag(b2)];
            z2imag = interp1(x2,y2imag,[2 3]);
            z62 = z2real(1) +1i*z2imag(1);
            z72 = z2real(2) +1i*z2imag(2);
            h(n,6+j) = z62;
            h(n,7+j) = z72;
            end

            end
        end

        h(:,columnas-1) = h(:,columnas-2);
        h(:,columnas) = h(:,columnas-2);

H1 = zeros(size(h));
[filas1 columnas1] = size(H1);
a = 4;
for i=1:(filas1-a)
    H1(i+a,:) = h(i,:);
end
for j=(filas1-a):filas1
    H1((j-filas1+a+1),:) = h(j,:);
end

h1 = zeros(GP,columnas);
H = [h1;H1;h1];

end
-------------------------------------------------------------------------------

function [salidaPSrx] = conversorPSrx2 (salidaEcualizador0,GP,Npilotos)

[filas0,columnas0] = size(salidaEcualizador0);
salidaEcualizador = salidaEcualizador0((GP+1):(filas0-GP),:);
[filas,columnas] = size(salidaEcualizador);
salidaPSrx = zeros(1,filas*columnas-Npilotos);

% Definimos la ubicación de las señales piloto
a = zeros(1,ceil(filas/6));
k = zeros(1,ceil(filas/6));
b = zeros(1,ceil(columnas/7));
l = zeros(1,ceil(columnas/7));
p = 1;

for i=0:6:(filas-1)
    for j=0:7:(columnas-1)
    aa = 3+i;
    a(1,p) = aa;
    c = 6+i;
    k(1,p) = c;
```

```
    bb = 5+j;
    b(1,p) = bb;
    d = 1+j;
    l(1,p) = d;
    p = p+1;
    end
end

q=1;
for n=1:columnas
   for o=1:filas
       % Recorremos la matriz con el objetivo de extraer la salida del
       % ecualizador que se trata de un vector lineal
       % Si llegamos a una posición de la matriz correspondiente a una
       % señal piloto no usamos esa posición en la salida del ecualizador, en caso
       % contrario introducimos el símbolo de tráfico del modulador en la
       % posición correspondiente de la salida del ecualizador

       for m=1:length(k)
            if (k(m)==o && l(m)==n)
                q = q-1;
                break;
            elseif (a(m)==o && b(m)==n)
                q = q-1;
                break;
            else
                salidaPSrx(q)=salidaEcualizador(o,n);
            end
       end
       q=q+1;
    end
end
end

--------------------------------------------------------------------------------

function [salidaDemodulador] = ElegirDemodulador(salidaPSrx,tipoConstelacion)

% Elegimos la constelación que queremos:
% 1 - Constelación QPSK
% 2 - Constelación 16QAM
% 3 - Constelación 64QAM

switch(tipoConstelacion)
    case 1
        salidaDemodulador = DemoduladorQPSK(salidaPSrx);
    case 2
        salidaDemodulador = Demodulador16QAM(salidaPSrx);
    case 3
        salidaDemodulador = Demodulador64QAM(salidaPSrx);
end
end

--------------------------------------------------------------------------------

function [salidaDemodulador] = DemoduladorQPSK (salidaPSrx)

salidaPSrx = salidaPSrx*sqrt(2); %Desnormalizamos
N = length(salidaPSrx);
salidaDemodulador = zeros(1,2*N);

aux = 1;
for n=1:N
    if (real(salidaPSrx(n))>0)
        if (imag(salidaPSrx(n))>0)
            salidaDemodulador(aux)=0;
```

```
            salidaDemodulador(aux+1)=0;
        else
            salidaDemodulador(aux)=0;
            salidaDemodulador(aux+1)=1;
        end
    else
        if (imag(salidaPSrx(n))>0)
            salidaDemodulador(aux)=1;
            salidaDemodulador(aux+1)=0;
        else
            salidaDemodulador(aux)=1;
            salidaDemodulador(aux+1)=1;
        end
    end
    aux = aux +2;
end
end

-------------------------------------------------------------------------------
function [salidaDemodulador] = Demodulador16QAM (salidaPSrx)

salidaPSrx = salidaPSrx*sqrt(10); %Desnormalizamos
N = length(salidaPSrx);
salidaDemodulador = zeros(1,4*N);

aux = 1;
for n=1:N
    if (real(salidaPSrx(n))>0)
        if (real(salidaPSrx(n))>2)
            if (imag(salidaPSrx(n))>0)
                if (imag(salidaPSrx(n))>2)
                    salidaDemodulador(aux)=0;
                    salidaDemodulador(aux+1)=0;
                    salidaDemodulador(aux+2)=1;
                    salidaDemodulador(aux+3)=1;
                else
                    salidaDemodulador(aux)=0;
                    salidaDemodulador(aux+1)=0;
                    salidaDemodulador(aux+2)=1;
                    salidaDemodulador(aux+3)=0;
                end
            else
                 if (imag(salidaPSrx(n))>-2)
                    salidaDemodulador(aux)=0;
                    salidaDemodulador(aux+1)=1;
                    salidaDemodulador(aux+2)=1;
                    salidaDemodulador(aux+3)=1;
                else
                    salidaDemodulador(aux)=0;
                    salidaDemodulador(aux+1)=1;
                    salidaDemodulador(aux+2)=1;
                    salidaDemodulador(aux+3)=0;
                 end
            end
        else
            if (imag(salidaPSrx(n))>0)
                if (imag(salidaPSrx(n))>2)
                    salidaDemodulador(aux)=0;
                    salidaDemodulador(aux+1)=0;
                    salidaDemodulador(aux+2)=0;
                    salidaDemodulador(aux+3)=1;
                else
                    salidaDemodulador(aux)=0;
                    salidaDemodulador(aux+1)=0;
                    salidaDemodulador(aux+2)=0;
                    salidaDemodulador(aux+3)=0;
```

```
            end
        else
             if (imag(salidaPSrx(n))>-2)
                salidaDemodulador(aux)=0;
                salidaDemodulador(aux+1)=1;
                salidaDemodulador(aux+2)=0;
                salidaDemodulador(aux+3)=1;
            else
                salidaDemodulador(aux)=0;
                salidaDemodulador(aux+1)=1;
                salidaDemodulador(aux+2)=0;
                salidaDemodulador(aux+3)=0;
             end
        end
    end
else
    if (real(salidaPSrx(n))>-2)
        if (imag(salidaPSrx(n))>0)
            if (imag(salidaPSrx(n))>2)
                salidaDemodulador(aux)=1;
                salidaDemodulador(aux+1)=0;
                salidaDemodulador(aux+2)=1;
                salidaDemodulador(aux+3)=1;
            else
                salidaDemodulador(aux)=1;
                salidaDemodulador(aux+1)=0;
                salidaDemodulador(aux+2)=1;
                salidaDemodulador(aux+3)=0;
            end
        else
             if (imag(salidaPSrx(n))>-2)
                salidaDemodulador(aux)=1;
                salidaDemodulador(aux+1)=1;
                salidaDemodulador(aux+2)=1;
                salidaDemodulador(aux+3)=1;
            else
                salidaDemodulador(aux)=1;
                salidaDemodulador(aux+1)=1;
                salidaDemodulador(aux+2)=1;
                salidaDemodulador(aux+3)=0;
             end
        end
    else
        if (imag(salidaPSrx(n))>0)
            if (imag(salidaPSrx(n))>2)
                salidaDemodulador(aux)=1;
                salidaDemodulador(aux+1)=0;
                salidaDemodulador(aux+2)=0;
                salidaDemodulador(aux+3)=1;
            else
                salidaDemodulador(aux)=1;
                salidaDemodulador(aux+1)=0;
                salidaDemodulador(aux+2)=0;
                salidaDemodulador(aux+3)=0;
            end
        else
             if (imag(salidaPSrx(n))>-2)
                salidaDemodulador(aux)=1;
                salidaDemodulador(aux+1)=1;
                salidaDemodulador(aux+2)=0;
                salidaDemodulador(aux+3)=1;
            else
                salidaDemodulador(aux)=1;
                salidaDemodulador(aux+1)=1;
                salidaDemodulador(aux+2)=0;
                salidaDemodulador(aux+3)=0;
```

```
                end
            end
        end
    end

    aux = aux +4;
end
end

--------------------------------------------------------------------------------

function [salidaDemodulador] = Demodulador64QAM (salidaPSrx)

salidaPSrx = salidaPSrx*sqrt(42); %Desnormalizamos
N = length(salidaPSrx);
salidaDemodulador = zeros(1,6*N);

aux = 1;
for n=1:N
    if (real(salidaPSrx(n))>0)
        if (real(salidaPSrx(n))>2)
            if (real(salidaPSrx(n))>4)
                if (real(salidaPSrx(n))>6)
                    if (imag(salidaPSrx(n))>0)
                        if (imag(salidaPSrx(n))>2)
                            if (imag(salidaPSrx(n))>4)
                                if (imag(salidaPSrx(n))>6)
                                    salidaDemodulador(aux)=0;
                                    salidaDemodulador(aux+1)=1;
                                    salidaDemodulador(aux+2)=1;
                                    salidaDemodulador(aux+3)=0;
                                    salidaDemodulador(aux+4)=1;
                                    salidaDemodulador(aux+5)=1;
                                else
                                    salidaDemodulador(aux)=0;
                                    salidaDemodulador(aux+1)=1;
                                    salidaDemodulador(aux+2)=1;
                                    salidaDemodulador(aux+3)=0;
                                    salidaDemodulador(aux+4)=1;
                                    salidaDemodulador(aux+5)=0;
                                end
                            else
                                salidaDemodulador(aux)=0;
                                salidaDemodulador(aux+1)=1;
                                salidaDemodulador(aux+2)=1;
                                salidaDemodulador(aux+3)=0;
                                salidaDemodulador(aux+4)=0;
                                salidaDemodulador(aux+5)=0;
                            end
                        else
                            salidaDemodulador(aux)=0;
                            salidaDemodulador(aux+1)=1;
                            salidaDemodulador(aux+2)=1;
                            salidaDemodulador(aux+3)=0;
                            salidaDemodulador(aux+4)=0;
                            salidaDemodulador(aux+5)=1;
                        end
                    else
                        if (imag(salidaPSrx(n))<-2)
                            if (imag(salidaPSrx(n))<-4)
                                if (imag(salidaPSrx(n))<-6)
                                    salidaDemodulador(aux)=0;
                                    salidaDemodulador(aux+1)=1;
                                    salidaDemodulador(aux+2)=1;
                                    salidaDemodulador(aux+3)=1;
                                    salidaDemodulador(aux+4)=1;
```

```
                        salidaDemodulador(aux+5)=1;
                    else
                        salidaDemodulador(aux)=0;
                        salidaDemodulador(aux+1)=1;
                        salidaDemodulador(aux+2)=1;
                        salidaDemodulador(aux+3)=1;
                        salidaDemodulador(aux+4)=1;
                        salidaDemodulador(aux+5)=0;
                    end
                else
                    salidaDemodulador(aux)=0;
                    salidaDemodulador(aux+1)=1;
                    salidaDemodulador(aux+2)=1;
                    salidaDemodulador(aux+3)=1;
                    salidaDemodulador(aux+4)=0;
                    salidaDemodulador(aux+5)=0;
                end
            else
                salidaDemodulador(aux)=0;
                salidaDemodulador(aux+1)=1;
                salidaDemodulador(aux+2)=1;
                salidaDemodulador(aux+3)=1;
                salidaDemodulador(aux+4)=0;
                salidaDemodulador(aux+5)=1;
            end
        end
    else
        if (imag(salidaPSrx(n))>0)
            if (imag(salidaPSrx(n))>2)
                if (imag(salidaPSrx(n))>4)
                    if (imag(salidaPSrx(n))>6)
                        salidaDemodulador(aux)=0;
                        salidaDemodulador(aux+1)=1;
                        salidaDemodulador(aux+2)=0;
                        salidaDemodulador(aux+3)=0;
                        salidaDemodulador(aux+4)=1;
                        salidaDemodulador(aux+5)=1;
                    else
                        salidaDemodulador(aux)=0;
                        salidaDemodulador(aux+1)=1;
                        salidaDemodulador(aux+2)=0;
                        salidaDemodulador(aux+3)=0;
                        salidaDemodulador(aux+4)=1;
                        salidaDemodulador(aux+5)=0;
                    end
                else
                    salidaDemodulador(aux)=0;
                    salidaDemodulador(aux+1)=1;
                    salidaDemodulador(aux+2)=0;
                    salidaDemodulador(aux+3)=0;
                    salidaDemodulador(aux+4)=0;
                    salidaDemodulador(aux+5)=0;
                end
            else
                salidaDemodulador(aux)=0;
                salidaDemodulador(aux+1)=1;
                salidaDemodulador(aux+2)=0;
                salidaDemodulador(aux+3)=0;
                salidaDemodulador(aux+4)=0;
                salidaDemodulador(aux+5)=1;
            end
        else
            if (imag(salidaPSrx(n))<-2)
                if (imag(salidaPSrx(n))<-4)
                    if (imag(salidaPSrx(n))<-6)
                        salidaDemodulador(aux)=0;
```

```
                            salidaDemodulador(aux+1)=1;
                            salidaDemodulador(aux+2)=0;
                            salidaDemodulador(aux+3)=1;
                            salidaDemodulador(aux+4)=1;
                            salidaDemodulador(aux+5)=1;
                        else
                            salidaDemodulador(aux)=0;
                            salidaDemodulador(aux+1)=1;
                            salidaDemodulador(aux+2)=0;
                            salidaDemodulador(aux+3)=1;
                            salidaDemodulador(aux+4)=1;
                            salidaDemodulador(aux+5)=0;
                        end
                    else
                        salidaDemodulador(aux)=0;
                        salidaDemodulador(aux+1)=1;
                        salidaDemodulador(aux+2)=0;
                        salidaDemodulador(aux+3)=1;
                        salidaDemodulador(aux+4)=0;
                        salidaDemodulador(aux+5)=0;
                    end
                else
                    salidaDemodulador(aux)=0;
                    salidaDemodulador(aux+1)=1;
                    salidaDemodulador(aux+2)=0;
                    salidaDemodulador(aux+3)=1;
                    salidaDemodulador(aux+4)=0;
                    salidaDemodulador(aux+5)=1;
                end
            end
        end
    else
        if (imag(salidaPSrx(n))>0)
                if (imag(salidaPSrx(n))>2)
                    if (imag(salidaPSrx(n))>4)
                        if (imag(salidaPSrx(n))>6)
                            salidaDemodulador(aux)=0;
                            salidaDemodulador(aux+1)=0;
                            salidaDemodulador(aux+2)=0;
                            salidaDemodulador(aux+3)=0;
                            salidaDemodulador(aux+4)=1;
                            salidaDemodulador(aux+5)=1;
                        else
                            salidaDemodulador(aux)=0;
                            salidaDemodulador(aux+1)=0;
                            salidaDemodulador(aux+2)=0;
                            salidaDemodulador(aux+3)=0;
                            salidaDemodulador(aux+4)=1;
                            salidaDemodulador(aux+5)=0;
                        end
                    else
                        salidaDemodulador(aux)=0;
                        salidaDemodulador(aux+1)=0;
                        salidaDemodulador(aux+2)=0;
                        salidaDemodulador(aux+3)=0;
                        salidaDemodulador(aux+4)=0;
                        salidaDemodulador(aux+5)=0;
                    end
                else
                    salidaDemodulador(aux)=0;
                    salidaDemodulador(aux+1)=0;
                    salidaDemodulador(aux+2)=0;
                    salidaDemodulador(aux+3)=0;
                    salidaDemodulador(aux+4)=0;
                    salidaDemodulador(aux+5)=1;
                end
```

```
            else
                if (imag(salidaPSrx(n))<-2)
                    if (imag(salidaPSrx(n))<-4)
                        if (imag(salidaPSrx(n))<-6)
                            salidaDemodulador(aux)=0;
                            salidaDemodulador(aux+1)=0;
                            salidaDemodulador(aux+2)=0;
                            salidaDemodulador(aux+3)=1;
                            salidaDemodulador(aux+4)=1;
                            salidaDemodulador(aux+5)=1;
                        else
                            salidaDemodulador(aux)=0;
                            salidaDemodulador(aux+1)=0;
                            salidaDemodulador(aux+2)=0;
                            salidaDemodulador(aux+3)=1;
                            salidaDemodulador(aux+4)=1;
                            salidaDemodulador(aux+5)=0;
                        end
                    else
                        salidaDemodulador(aux)=0;
                        salidaDemodulador(aux+1)=0;
                        salidaDemodulador(aux+2)=0;
                        salidaDemodulador(aux+3)=1;
                        salidaDemodulador(aux+4)=0;
                        salidaDemodulador(aux+5)=0;
                    end
                else
                    salidaDemodulador(aux)=0;
                    salidaDemodulador(aux+1)=0;
                    salidaDemodulador(aux+2)=0;
                    salidaDemodulador(aux+3)=1;
                    salidaDemodulador(aux+4)=0;
                    salidaDemodulador(aux+5)=1;
                end
        end
    end
else
    if (imag(salidaPSrx(n))>0)
                if (imag(salidaPSrx(n))>2)
                    if (imag(salidaPSrx(n))>4)
                        if (imag(salidaPSrx(n))>6)
                            salidaDemodulador(aux)=0;
                            salidaDemodulador(aux+1)=0;
                            salidaDemodulador(aux+2)=1;
                            salidaDemodulador(aux+3)=0;
                            salidaDemodulador(aux+4)=1;
                            salidaDemodulador(aux+5)=1;
                        else
                            salidaDemodulador(aux)=0;
                            salidaDemodulador(aux+1)=0;
                            salidaDemodulador(aux+2)=1;
                            salidaDemodulador(aux+3)=0;
                            salidaDemodulador(aux+4)=1;
                            salidaDemodulador(aux+5)=0;
                        end
                    else
                        salidaDemodulador(aux)=0;
                        salidaDemodulador(aux+1)=0;
                        salidaDemodulador(aux+2)=1;
                        salidaDemodulador(aux+3)=0;
                        salidaDemodulador(aux+4)=0;
                        salidaDemodulador(aux+5)=0;
                    end
                else
                    salidaDemodulador(aux)=0;
                    salidaDemodulador(aux+1)=0;
```

```
                        salidaDemodulador(aux+2)=1;
                        salidaDemodulador(aux+3)=0;
                        salidaDemodulador(aux+4)=0;
                        salidaDemodulador(aux+5)=1;
                    end
                else
                    if (imag(salidaPSrx(n))<-2)
                        if (imag(salidaPSrx(n))<-4)
                            if (imag(salidaPSrx(n))<-6)
                                salidaDemodulador(aux)=0;
                                salidaDemodulador(aux+1)=0;
                                salidaDemodulador(aux+2)=1;
                                salidaDemodulador(aux+3)=1;
                                salidaDemodulador(aux+4)=1;
                                salidaDemodulador(aux+5)=1;
                            else
                                salidaDemodulador(aux)=0;
                                salidaDemodulador(aux+1)=0;
                                salidaDemodulador(aux+2)=1;
                                salidaDemodulador(aux+3)=1;
                                salidaDemodulador(aux+4)=1;
                                salidaDemodulador(aux+5)=0;
                            end
                        else
                            salidaDemodulador(aux)=0;
                            salidaDemodulador(aux+1)=0;
                            salidaDemodulador(aux+2)=1;
                            salidaDemodulador(aux+3)=1;
                            salidaDemodulador(aux+4)=0;
                            salidaDemodulador(aux+5)=0;
                        end
                    else
                        salidaDemodulador(aux)=0;
                        salidaDemodulador(aux+1)=0;
                        salidaDemodulador(aux+2)=1;
                        salidaDemodulador(aux+3)=1;
                        salidaDemodulador(aux+4)=0;
                        salidaDemodulador(aux+5)=1;
                    end
            end
        end
    else
        if (real(salidaPSrx(n))<-2)
            if (real(salidaPSrx(n))<-4)
                if (real(salidaPSrx(n))<-6)
                    if (imag(salidaPSrx(n))>0)
                        if (imag(salidaPSrx(n))>2)
                            if (imag(salidaPSrx(n))>4)
                                if (imag(salidaPSrx(n))>6)
                                    salidaDemodulador(aux)=1;
                                    salidaDemodulador(aux+1)=1;
                                    salidaDemodulador(aux+2)=1;
                                    salidaDemodulador(aux+3)=0;
                                    salidaDemodulador(aux+4)=1;
                                    salidaDemodulador(aux+5)=1;
                                else
                                    salidaDemodulador(aux)=1;
                                    salidaDemodulador(aux+1)=1;
                                    salidaDemodulador(aux+2)=1;
                                    salidaDemodulador(aux+3)=0;
                                    salidaDemodulador(aux+4)=1;
                                    salidaDemodulador(aux+5)=0;
                                end
                            else
                                salidaDemodulador(aux)=1;
                                salidaDemodulador(aux+1)=1;
```

```
                    salidaDemodulador(aux+2)=1;
                    salidaDemodulador(aux+3)=0;
                    salidaDemodulador(aux+4)=0;
                    salidaDemodulador(aux+5)=0;
                end
            else
                salidaDemodulador(aux)=1;
                salidaDemodulador(aux+1)=1;
                salidaDemodulador(aux+2)=1;
                salidaDemodulador(aux+3)=0;
                salidaDemodulador(aux+4)=0;
                salidaDemodulador(aux+5)=1;
            end
        else
            if (imag(salidaPSrx(n))<-2)
                if (imag(salidaPSrx(n))<-4)
                    if (imag(salidaPSrx(n))<-6)
                        salidaDemodulador(aux)=1;
                        salidaDemodulador(aux+1)=1;
                        salidaDemodulador(aux+2)=1;
                        salidaDemodulador(aux+3)=1;
                        salidaDemodulador(aux+4)=1;
                        salidaDemodulador(aux+5)=1;
                    else
                        salidaDemodulador(aux)=1;
                        salidaDemodulador(aux+1)=1;
                        salidaDemodulador(aux+2)=1;
                        salidaDemodulador(aux+3)=1;
                        salidaDemodulador(aux+4)=1;
                        salidaDemodulador(aux+5)=0;
                    end
                else
                    salidaDemodulador(aux)=1;
                    salidaDemodulador(aux+1)=1;
                    salidaDemodulador(aux+2)=1;
                    salidaDemodulador(aux+3)=1;
                    salidaDemodulador(aux+4)=0;
                    salidaDemodulador(aux+5)=0;
                end
            else
                salidaDemodulador(aux)=1;
                salidaDemodulador(aux+1)=1;
                salidaDemodulador(aux+2)=1;
                salidaDemodulador(aux+3)=1;
                salidaDemodulador(aux+4)=0;
                salidaDemodulador(aux+5)=1;
            end
        end
    else
        if (imag(salidaPSrx(n))>0)
            if (imag(salidaPSrx(n))>2)
                if (imag(salidaPSrx(n))>4)
                    if (imag(salidaPSrx(n))>6)
                        salidaDemodulador(aux)=1;
                        salidaDemodulador(aux+1)=1;
                        salidaDemodulador(aux+2)=0;
                        salidaDemodulador(aux+3)=0;
                        salidaDemodulador(aux+4)=1;
                        salidaDemodulador(aux+5)=1;
                    else
                        salidaDemodulador(aux)=1;
                        salidaDemodulador(aux+1)=1;
                        salidaDemodulador(aux+2)=0;
                        salidaDemodulador(aux+3)=0;
                        salidaDemodulador(aux+4)=1;
                        salidaDemodulador(aux+5)=0;
```

```
                    end
                else
                    salidaDemodulador(aux)=1;
                    salidaDemodulador(aux+1)=1;
                    salidaDemodulador(aux+2)=0;
                    salidaDemodulador(aux+3)=0;
                    salidaDemodulador(aux+4)=0;
                    salidaDemodulador(aux+5)=0;
                end
            else
                salidaDemodulador(aux)=1;
                salidaDemodulador(aux+1)=1;
                salidaDemodulador(aux+2)=0;
                salidaDemodulador(aux+3)=0;
                salidaDemodulador(aux+4)=0;
                salidaDemodulador(aux+5)=1;
            end
        else
            if (imag(salidaPSrx(n))<-2)
                if (imag(salidaPSrx(n))<-4)
                    if (imag(salidaPSrx(n))<-6)
                        salidaDemodulador(aux)=1;
                        salidaDemodulador(aux+1)=1;
                        salidaDemodulador(aux+2)=0;
                        salidaDemodulador(aux+3)=1;
                        salidaDemodulador(aux+4)=1;
                        salidaDemodulador(aux+5)=1;
                    else
                        salidaDemodulador(aux)=1;
                        salidaDemodulador(aux+1)=1;
                        salidaDemodulador(aux+2)=0;
                        salidaDemodulador(aux+3)=1;
                        salidaDemodulador(aux+4)=1;
                        salidaDemodulador(aux+5)=0;
                    end
                else
                    salidaDemodulador(aux)=1;
                    salidaDemodulador(aux+1)=1;
                    salidaDemodulador(aux+2)=0;
                    salidaDemodulador(aux+3)=1;
                    salidaDemodulador(aux+4)=0;
                    salidaDemodulador(aux+5)=0;
                end
            else
                salidaDemodulador(aux)=1;
                salidaDemodulador(aux+1)=1;
                salidaDemodulador(aux+2)=0;
                salidaDemodulador(aux+3)=1;
                salidaDemodulador(aux+4)=0;
                salidaDemodulador(aux+5)=1;
            end
        end
    end
else
    if (imag(salidaPSrx(n))>0)
            if (imag(salidaPSrx(n))>2)
                if (imag(salidaPSrx(n))>4)
                    if (imag(salidaPSrx(n))>6)
                        salidaDemodulador(aux)=1;
                        salidaDemodulador(aux+1)=0;
                        salidaDemodulador(aux+2)=0;
                        salidaDemodulador(aux+3)=0;
                        salidaDemodulador(aux+4)=1;
                        salidaDemodulador(aux+5)=1;
                    else
                        salidaDemodulador(aux)=1;
```

```
                                salidaDemodulador(aux+1)=0;
                                salidaDemodulador(aux+2)=0;
                                salidaDemodulador(aux+3)=0;
                                salidaDemodulador(aux+4)=1;
                                salidaDemodulador(aux+5)=0;
                            end
                        else
                            salidaDemodulador(aux)=1;
                            salidaDemodulador(aux+1)=0;
                            salidaDemodulador(aux+2)=0;
                            salidaDemodulador(aux+3)=0;
                            salidaDemodulador(aux+4)=0;
                            salidaDemodulador(aux+5)=0;
                        end
                    else
                        salidaDemodulador(aux)=1;
                        salidaDemodulador(aux+1)=0;
                        salidaDemodulador(aux+2)=0;
                        salidaDemodulador(aux+3)=0;
                        salidaDemodulador(aux+4)=0;
                        salidaDemodulador(aux+5)=1;
                    end
                else
                    if (imag(salidaPSrx(n))<-2)
                        if (imag(salidaPSrx(n))<-4)
                            if (imag(salidaPSrx(n))<-6)
                                salidaDemodulador(aux)=1;
                                salidaDemodulador(aux+1)=0;
                                salidaDemodulador(aux+2)=0;
                                salidaDemodulador(aux+3)=1;
                                salidaDemodulador(aux+4)=1;
                                salidaDemodulador(aux+5)=1;
                            else
                                salidaDemodulador(aux)=1;
                                salidaDemodulador(aux+1)=0;
                                salidaDemodulador(aux+2)=0;
                                salidaDemodulador(aux+3)=1;
                                salidaDemodulador(aux+4)=1;
                                salidaDemodulador(aux+5)=0;
                            end
                        else
                            salidaDemodulador(aux)=1;
                            salidaDemodulador(aux+1)=0;
                            salidaDemodulador(aux+2)=0;
                            salidaDemodulador(aux+3)=1;
                            salidaDemodulador(aux+4)=0;
                            salidaDemodulador(aux+5)=0;
                        end
                    else
                        salidaDemodulador(aux)=1;
                        salidaDemodulador(aux+1)=0;
                        salidaDemodulador(aux+2)=0;
                        salidaDemodulador(aux+3)=1;
                        salidaDemodulador(aux+4)=0;
                        salidaDemodulador(aux+5)=1;
                    end
            end
        end
    else
        if (imag(salidaPSrx(n))>0)
                    if (imag(salidaPSrx(n))>2)
                        if (imag(salidaPSrx(n))>4)
                            if (imag(salidaPSrx(n))>6)
                                salidaDemodulador(aux)=1;
                                salidaDemodulador(aux+1)=0;
                                salidaDemodulador(aux+2)=1;
```

```
                                salidaDemodulador(aux+3)=0;
                                salidaDemodulador(aux+4)=1;
                                salidaDemodulador(aux+5)=1;
                            else
                                salidaDemodulador(aux)=1;
                                salidaDemodulador(aux+1)=0;
                                salidaDemodulador(aux+2)=1;
                                salidaDemodulador(aux+3)=0;
                                salidaDemodulador(aux+4)=1;
                                salidaDemodulador(aux+5)=0;
                            end
                        else
                            salidaDemodulador(aux)=1;
                            salidaDemodulador(aux+1)=0;
                            salidaDemodulador(aux+2)=1;
                            salidaDemodulador(aux+3)=0;
                            salidaDemodulador(aux+4)=0;
                            salidaDemodulador(aux+5)=0;
                        end
                    else
                        salidaDemodulador(aux)=1;
                        salidaDemodulador(aux+1)=0;
                        salidaDemodulador(aux+2)=1;
                        salidaDemodulador(aux+3)=0;
                        salidaDemodulador(aux+4)=0;
                        salidaDemodulador(aux+5)=1;
                    end
                else
                    if (imag(salidaPSrx(n))<-2)
                        if (imag(salidaPSrx(n))<-4)
                            if (imag(salidaPSrx(n))<-6)
                                salidaDemodulador(aux)=1;
                                salidaDemodulador(aux+1)=0;
                                salidaDemodulador(aux+2)=1;
                                salidaDemodulador(aux+3)=1;
                                salidaDemodulador(aux+4)=1;
                                salidaDemodulador(aux+5)=1;
                            else
                                salidaDemodulador(aux)=1;
                                salidaDemodulador(aux+1)=0;
                                salidaDemodulador(aux+2)=1;
                                salidaDemodulador(aux+3)=1;
                                salidaDemodulador(aux+4)=1;
                                salidaDemodulador(aux+5)=0;
                            end
                        else
                            salidaDemodulador(aux)=1;
                            salidaDemodulador(aux+1)=0;
                            salidaDemodulador(aux+2)=1;
                            salidaDemodulador(aux+3)=1;
                            salidaDemodulador(aux+4)=0;
                            salidaDemodulador(aux+5)=0;
                        end
                    else
                        salidaDemodulador(aux)=1;
                        salidaDemodulador(aux+1)=0;
                        salidaDemodulador(aux+2)=1;
                        salidaDemodulador(aux+3)=1;
                        salidaDemodulador(aux+4)=0;
                        salidaDemodulador(aux+5)=1;
                    end
            end
        end
```

```
    end
aux = aux +6;
end
```

CÓDIGO DEL SISTEMA MIMO 2x2

Archivos necesarios:

- anadirPrefijoCiclico.m
- conversorPSrxMIMOAntena1
- conversorPSrxMIMOAntena2
- conversorPStx.m
- conversorSPrx.m
- conversorSPtxAntena1MaximosPilotos.m
- conversorSPtxAntena1PilotosEstandar.m
- conversorSPtxAntena2MaximosPilotos.m
- conversorSPtxAntena2PilotosEstandar.m
- Demodulador16QAM.m
- Demodulador64QAM.m
- DemoduladorQPSK.m
- ecualizadorMatrizEnteraMIMO2Antena1.m
- ecualizadorMatrizEnteraMIMO2Antena2.m
- ElegirDemodulador.m
- ElegirModulador.m
- eliminarPrefijoCiclico.m
- fftrxMIMO.m
- iffttxMIMO.m
- MIMO2x2.m
- Modulador16QAM.m
- Modulador64QAM.m
- ModuladorQPSK.m

A continuación se muestra el script del código elaborado para el sistema MIMO 2x2 (Mimo2x2.m) con dos antenas transmisoras y con dos antenas receptoras.
Posteriormente se muestran el resto de funciones necesarias, exceptuando aquellas que son las mismas funciones en el sistema SISO y que fueron definidas anteriormente.

```
clc
clear all
close all
```

```
% MIMO: 2 antenas en el tx y 2 en el rx

% Bits de entrada que queremos enviar
bitsEntrada = round(rand(1,10000));

% Repartimos los bits de entrada para cada una de las antenas
bitsEntradaAntena1 = bitsEntrada(1:length(bitsEntrada)/2);
bitsEntradaAntena2 = bitsEntrada((length(bitsEntrada)/2+1):end);

% 1 - TRANSMISOR

% 1.1 - Modulador
% Convertimos los bits de entrada en símbolos
% Elegimos la constelación que queremos mediante la variable tipoConstelacion:
% 1 - Constelación QPSK
% 2 - Constelación 16QAM
% 3 - Constelación 64QAM
tipoConstelacion = 1;
salidaModuladorAntena1 = ElegirModulador(bitsEntradaAntena1,tipoConstelacion);
PotenciaSenalAntena1 = var(salidaModuladorAntena1);
salidaModuladorAntena2 = ElegirModulador(bitsEntradaAntena2,tipoConstelacion);
PotenciaSenalAntena2 = var(salidaModuladorAntena2);

% 1.2 - Conversor Serie Paralelo

Nfft = 512; % Definimos el tamaño de la FFT
P = 1; % Valor de la señal piloto

% Se usan las frecuencias útiles definidas por el estándar
[GP,Npilotos1,salidaSPtxAntena1] =
conversorSPtxAntena1PilotosEstandar(salidaModuladorAntena1,Nfft,P);
[GP1,Npilotos2,salidaSPtxAntena2] =
conversorSPtxAntena2PilotosEstandar(salidaModuladorAntena2,Nfft,P);

% Se usan el máximo número de pilotos posibles
% [GP,Npilotos1,salidaSPtxAntena1] =
conversorSPtxAntena1MaximosPilotos(salidaModuladorAntena1,Nfft,P);
% [GP,Npilotos2,salidaSPtxAntena2] =
conversorSPtxAntena2MaximosPilotos(salidaModuladorAntena2,Nfft,P);

[tam1,tam2] = size(salidaSPtxAntena1);
salidaSPtx = zeros(tam1,tam2,2);
salidaSPtx(:,:,1) = salidaSPtxAntena1;
salidaSPtx(:,:,2) = salidaSPtxAntena2;

% 1.3 - IFFT
%Pasamos del dominio de la frecuencia al dominio del tiempo

salidaifft = iffttxMIMO(salidaSPtx,Nfft);

% 1.4 - Añadir prefijo cíclico
% La adición del prefijo cíclico consiste en copiar la parte final del
% símbolo al principio del mismo, es decir, el final de la columna del símbolo
% correspondiente, en el principio de la misma

tamPC = Nfft/8; % Tamaño del prefijo cíclico

[filas,columnas,antenas] = size(salidaifft);
salidaPCtx = zeros(filas+tamPC,columnas);

for m=1:antenas
    for n=1:columnas
        salidaPCtx(:,n,m) = anadirPrefijoCiclico(salidaifft(:,n,m),tamPC);
    end
end
```

```
% 1.5 - Conversor Paralelo Serie

salidaPStx1 = conversorPStx(salidaPCtx(:,:,1));
salidaPStx2 = conversorPStx(salidaPCtx(:,:,2));

% 2 - CANAL
% Definimos el tipo de canal existente entre el transmisor y el receptor

ni = 0.52;
ni1 =
(ni)*(1/sqrt(2))*(randn(1,length(salidaPStx1))+1i*randn(1,length(salidaPStx1)));
ni2 =
(ni)*(1/sqrt(2))*(randn(1,length(salidaPStx2))+1i*randn(1,length(salidaPStx2)));
PotenciaRuido1 = var(ni1);
PotenciaRuido2 = var(ni2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
canal11=[0.909505559686059+0.0570811522831248i
0.0456760830213736+0.111242422458266i -0.247602286506910-0.009283304416878O5i
0.0610869817003726-0.0342405687499177i -0.008029148864771 59+0.0224937836985498i
0.153191551329004+0.130831812483216i];
salidaCanal11 = filter(canal11, 1, salidaPStx1);

canal12=[0.566746906538893+0.630113834041522i 0.0357449752583050-
0.0679886475792722i 0.270114445438092+0.183711628767796i -
0.0778588683297167+0.006747986835070861 -0.025279032918100 5-0.0356571054464926i
0.132353771086039+0.157961257698125i]; % dos caminos
salidaCanal12 = filter(canal12, 1, salidaPStx2);

salidaCanal1 = salidaCanal11+salidaCanal12;

PotenciaSenal1 = var(salidaCanal1);
norm1 = PotenciaSenalAntena1/PotenciaSenal1;
salidaCanal1 = salidaCanal1.*sqrt(norm1);
PotenciaSenal1 = var(salidaCanal1);
salidaCanal1ConRuido = salidaCanal1+ni1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
canal22=[-0.531443401493595+0.242231727705288i -
0.145146758621639+0.0783346111385592i -0.116511995739920-0.0220821013769710i
0.0270894999888697-0.0403006766054490i 0.028718771505043 0+0.0777286940040754i
0.202029013372210-0.0614940293559058i];
salidaCanal22 = filter(canal22, 1, salidaPStx2);

canal21=[-0.474666676819643+0.358708495851864i 0.0406971799009435-
0.130806647107331i 0.0340745753950306-0.0121605503030910i -
0.008957530344932961+0.0408387125448647i -0.0735779229265508-0.0488901056490232i
0.0931540690409374-0.0856279555002441i]; % dos caminos
salidaCanal21 = filter(canal21, 1, salidaPStx1);

salidaCanal2 = salidaCanal22+salidaCanal21;
PotenciaSenal2 = var(salidaCanal2);
norm2 = PotenciaSenalAntena2/PotenciaSenal2;
salidaCanal2 = salidaCanal2.*sqrt(norm2);
PotenciaSenal2 = var(salidaCanal2);
salidaCanal2ConRuido = salidaCanal2+ni2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SNR1dB = 10*log10(PotenciaSenal1/PotenciaRuido1);
SNR2dB = 10*log10(PotenciaSenal2/PotenciaRuido2);

% 3 - RECEPTOR

% 3.1 - Conversor Serie Paralelo

salidaSPrx1 = conversorSPrx(salidaCanal1ConRuido,Nfft,tamPC);
```

```
salidaSPrx2 = conversorSPrx(salidaCanal2ConRuido,Nfft,tamPC);
[filas,columnas] = size(salidaSPrx1);
salidaSPrx = zeros(filas,columnas,2);
salidaSPrx(:,:,1) = salidaSPrx1(:,:);
salidaSPrx(:,:,2) = salidaSPrx2(:,:);

% 3.2 - Eliminar Prefijo Cíclico
% Eliminamos el comienzo de cada una de los columnas
% El tamaño del prefijo cíclico está definido anteriormente en tamPC

[filas,columnas,antenas] = size(salidaSPrx);
salidaPCrx = zeros(filas-tamPC,columnas,antenas);

for m=1:antenas
    for n=1:columnas
       salidaPCrx(:,n,m) = eliminarPrefijoCiclico(salidaSPrx(:,n,m),tamPC);
    end
end

% 3.1 - FFT
% Pasamos del dominio del tiempo al dominio frecuencial

salidafft = fftrxMIMO(salidaPCrx,Nfft);
[ff,cf,af] = size(salidafft);
salidafft1 = zeros(ff,cf);
salidafft1(:,:) = salidafft(:,:,1);
salidafft2 = zeros(ff,cf);
salidafft2(:,:) = salidafft(:,:,2);

% 3.2 - Ecualizador
% Revierte el impacto del canal multiplicando cada subportadora por un número
% complejo basado en la estimación del canal

% Obtenemos la matriz H que será la encargada de indicarnos la variación de
% amplitud y fase que ha sufrido cada uno de los elementos de los
% diferentes símbolos

H11 = ecualizadorMatrizEnteraMIMO2Antena1(salidafft(:,:,1),P,GP);
H12 = ecualizadorMatrizEnteraMIMO2Antena2(salidafft(:,:,1),P,GP);
H21 = ecualizadorMatrizEnteraMIMO2Antena1(salidafft(:,:,2),P,GP);
H22 = ecualizadorMatrizEnteraMIMO2Antena2(salidafft(:,:,2),P,GP);
[filasH,columnasH] = size(H11);

salidaEcualizador = zeros(filasH,columnasH,2);

for i=1:filasH
    for j=1:columnasH
        H = [H11(i,j) H12(i,j); H21(i,j) H22(i,j)];
        x = [salidafft(i,j,1) ; salidafft(i,j,2)];
        z = pinv(H)*x;
        salidaEcualizador(i,j,1) = z(1,1);
        salidaEcualizador(i,j,2) = z(2,1);
    end
end

% Conversor Paralelo Serie

salidaPSrxAntena1 =
conversorPSrxMIMOAntena1(salidaEcualizador(:,:,1),GP,Npilotos1);
salidaPSrxAntena2 =
conversorPSrxMIMOAntena1(salidaEcualizador(:,:,2),GP,Npilotos2);

% Demodulador
% Convierte los símbolos en una cadena de bits
% tipoConstelacion ya esta definido al comienzo de la tranmisión
```

```
salidaDemodulador1 =  ElegirDemodulador(salidaPSrxAntena1,tipoConstelacion);
salidaDemoduladorAntena1 = salidaDemodulador1(1:length(bitsEntradaAntena1));
salidaDemodulador2 =  ElegirDemodulador(salidaPSrxAntena2,tipoConstelacion);
salidaDemoduladorAntena2 = salidaDemodulador2(1:length(bitsEntradaAntena2));

% La salida se corresponderá con una cadena de bits del mismo tamaño que la
% enviada a través del transmisor
% Calculamos el número de fallos y el porcentaje de los mismos respecto a
% los bits de entrada y los bits de salida obtenidos
fallos1 = 0;
for i=1:length(bitsEntradaAntena1)
    b1 = (abs(salidaDemoduladorAntena1(i)-bitsEntradaAntena1(i)));
    fallos1 = fallos1+b1;
end
PorcentajeFallosAntena1 = (fallos1*100)/length(bitsEntradaAntena1);
fallos2 = 0;
for i=1:length(bitsEntradaAntena2)
    b2 = (abs(salidaDemoduladorAntena2(i)-bitsEntradaAntena2(i)));
    fallos2 = fallos2+b2;
end
PorcentajeFallosAntena2 = (fallos2*100)/length(bitsEntradaAntena2);
SNRdB = (SNR1dB+SNR2dB)/2;
PorcentajeFallos = (PorcentajeFallosAntena1+PorcentajeFallosAntena2)/2;

--------------------------------------------------------------------------------

function [GP,Npilotos,salidaSPtx] = conversorSPtxAntena1PilotosEstandar
(salidaModulador,Nfft,P)

% Convierte los símbolos que salen en serie del modulador en una matriz,
% cuyas filas se corresponden con las diferentes frecuencias portadoras y
% cuyas columnas se corresponden con los slots de tiempo.
% El número de filas vendrá definido por el valor de Nfft y existirán unas
% frecuencias de guarda.
% Se insertaran las señales piloto (de valor P) que servirán para la
% estimación del canal en el ecualizador
% Cada símbolo es una columna completa definida en un slot de tiempo
% concreto

n = length(salidaModulador);
Npilotos = floor(8*n/(12*14));
Nceros = Npilotos;
filas = Nfft;
switch (Nfft)
    case 128
        GP = (Nfft-72)/2;
    case 256
        GP = (Nfft-180)/2;
    case 512
        GP = (Nfft-300)/2;
    case 1024
        GP = (Nfft-600)/2;
    case 1536
        GP = (Nfft-900)/2;
    case 2048
        GP = (Nfft-1200)/2;
end
columnas = ceil((n+Npilotos+Nceros)/(filas-2*GP))+1;
salidaSPtx = zeros(filas,columnas);

% Definimos la ubicación de las señales piloto
a1 = zeros(1,ceil(filas/6));
k1 = zeros(1,ceil(filas/6));
b1 = zeros(1,ceil(columnas/7));
l1 = zeros(1,ceil(columnas/7));
p1 = 1;
```

```
for i=(GP+0):6:(filas+0-GP)
    for j=0:7:columnas

        aa1 = 3+i;
        a1(1,p1) = aa1;

        c1 = 6+i;
        k1(1,p1) = c1;

        bb1 = 5+j;
        b1(1,p1) = bb1;

        d1 = 1+j;
        l1(1,p1) = d1;

        p1 = p1+1;

    end
end

% Definimos las posiciones no usadas
a2 = zeros(1,ceil(filas/6));
k2 = zeros(1,ceil(filas/6));
b2 = zeros(1,ceil(columnas/7));
l2 = zeros(1,ceil(columnas/7));
p2 = 1;

for i=(GP+0):6:(filas+0-GP)
    for j=0:7:columnas

        aa2 = 3+i;
        a2(1,p2) = aa2;

        c2 = 6+i;
        k2(1,p2) = c2;

        bb2 = 1+j;
        b2(1,p2) = bb2;

        d2 = 5+j;
        l2(1,p2) = d2;

        p2 = p2+1;

    end
end


% Relleno de la matriz con el tráfico
% y con las señales piloto de referencia y las posiciones sin usar

q = 1;
for p=1:columnas
    for o=(GP+1):(filas-GP)
        % Recorremos la matriz
        % Si llegamos a una posición de la matriz correspondiente a una
        % señal piloto rellenamos la posición con el valor P, en caso
        % contrario introducimos el símbolo de tráfico del modulador que
        % corresponda

        for m=1:length(k1)
            if ((k1(m)==o && l1(m)==p))
                salidaSPtx(o,p)=P;
                q = q-1;
                break;
```

```
            elseif (a1(m)==o && b1(m)==p)
                salidaSPtx(o,p)=P;
                q = q-1;
                break;
            elseif ((k2(m)==o && l2(m)==p))
                salidaSPtx(o,p)=0;
                q = q-1;
                break;
            elseif (a2(m)==o && b2(m)==p)
                salidaSPtx(o,p)=0;
                q = q-1;
                break;
            else
                salidaSPtx(o,p)=salidaModulador(q);
            end
        end
        q = q+1;
        if (q==(length(salidaModulador)+1))
            break;
        end
    end
    if (q==(length(salidaModulador)+1))
        break;
    end
end



% Rellenamos con 0´s hasta que el número de columnas totales sea múltiplo
% de 7
columnas0 = 7*ceil(columnas/7) - columnas;
salidaSPtx = [salidaSPtx zeros(filas,columnas0)];
[filas1,columnas1] = size(salidaSPtx);

a1 = zeros(1,ceil(filas1/6));
k1 = zeros(1,ceil(filas1/6));
b1 = zeros(1,ceil(columnas1/7));
l1 = zeros(1,ceil(columnas1/7));
r1 = 1;

for m=(GP+0):6:(filas1+0-GP)
    for n=0:7:columnas1

        aa1 = 3+m;
        a1(1,r1) = aa1;

        c1 = 6+m;
        k1(1,r1) = c1;

        bb1 = 5+n;
        b1(1,r1) = bb1;

        d1 = 1+n;
        l1(1,r1) = d1;

        r1 = r1+1;

    end
end

a2 = zeros(1,ceil(filas1/6));
k2 = zeros(1,ceil(filas1/6));
b2 = zeros(1,ceil(columnas1/7));
l2 = zeros(1,ceil(columnas1/7));
r2 = 1;
```

```
for m=(GP+0):6:(filas1+0-GP)
    for n=0:7:columnas1

        aa2 = 3+m;
        a2(1,r2) = aa2;

        c2 = 6+m;
        k2(1,r2) = c2;

        bb2 = 5+n;
        b2(1,r2) = bb2;

        d2 = 1+n;
        l2(1,r2) = d2;

        r2 = r2+1;

    end
end


for p=1:columnas1
    for o=(GP+1):(filas-GP)

        for m=1:length(k1)
            if ((k1(m)==o && l1(m)==p))
                salidaSPtx(o,p)=P;
                break;
            elseif (a1(m)==o && b1(m)==p)
                salidaSPtx(o,p)=P;
                break;
            elseif ((k2(m)==o && l2(m)==p))
                salidaSPtx(o,p)=0;
                break;
            elseif (a2(m)==o && b2(m)==p)
                salidaSPtx(o,p)=0;
                break;
            else
                salidaSPtx(o,p)=salidaSPtx(o,p);
            end
        end
    end
end
end

--------------------------------------------------------------------------------

function [GP,Npilotos,salidaSPtx] = conversorSPtxAntena2PilotosEstandar
(salidaModulador,Nfft,P)

% Convierte los símbolos que salen en serie del modulador en una matriz,
% cuyas filas se corresponden con las diferentes frecuencias portadoras y
% cuyas columnas se corresponden con los slots de tiempo.
% El número de filas vendrá definido por el valor de Nfft y existirán unas
% frecuencias de guarda.
% Se insertaran las señales piloto (de valor P) que servirán para la
% estimación del canal en el ecualizador
% Cada símbolo es una columna completa definida en un slot de tiempo
% concreto

n = length(salidaModulador);
Npilotos = floor(8*n/(12*14));
Nceros = Npilotos;
filas = Nfft;
switch (Nfft)
```

```
    case 128
        GP = (Nfft-72)/2;
    case 256
        GP = (Nfft-180)/2;
    case 512
        GP = (Nfft-300)/2;
    case 1024
        GP = (Nfft-600)/2;
    case 1536
        GP = (Nfft-900)/2;
    case 2048
        GP = (Nfft-1200)/2;
end
columnas = ceil((n+Npilotos+Nceros)/(filas-2*GP))+1;
salidaSPtx = zeros(filas,columnas);

% Definimos la ubicación de las señales piloto
a1 = zeros(1,ceil(filas/6));
k1 = zeros(1,ceil(filas/6));
b1 = zeros(1,ceil(columnas/7));
l1 = zeros(1,ceil(columnas/7));
p1 = 1;

for i=(GP+0):6:(filas+0-GP)
    for j=0:7:columnas

        aa1 = 6+i;
        a1(1,p1) = aa1;

        c1 = 3+i;
        k1(1,p1) = c1;

        bb1 = 5+j;
        b1(1,p1) = bb1;

        d1 = 1+j;
        l1(1,p1) = d1;

        p1 = p1+1;

    end
end

% Definimos las posiciones no usadas
a2 = zeros(1,ceil(filas/6));
k2 = zeros(1,ceil(filas/6));
b2 = zeros(1,ceil(columnas/7));
l2 = zeros(1,ceil(columnas/7));
p2 = 1;

for i=(GP+0):6:(filas+0-GP)
    for j=0:7:columnas

        aa2 = 6+i;
        a2(1,p2) = aa2;

        c2 = 3+i;
        k2(1,p2) = c2;

        bb2 = 1+j;
        b2(1,p2) = bb2;

        d2 = 5+j;
        l2(1,p2) = d2;

        p2 = p2+1;
```

```
    end
end


% Relleno de la matriz con el tráfico
% y con las señales piloto de referencia y las posiciones sin usar

q = 1;
for p=1:columnas
    for o=(GP+1):(filas-GP)
        % Recorremos la matriz
        % Si llegamos a una posición de la matriz correspondiente a una
        % señal piloto rellenamos la posición con el valor P, en caso
        % contrario introducimos el símbolo de tráfico del modulador que
        % corresponda

        for m=1:length(k1)
            if ((k1(m)==o && l1(m)==p))
                salidaSPtx(o,p)=0;
                q = q-1;
                break;
            elseif (a1(m)==o && b1(m)==p)
                salidaSPtx(o,p)=0;
                q = q-1;
                break;
            elseif ((k2(m)==o && l2(m)==p))
                salidaSPtx(o,p)=P;
                q = q-1;
                break;
            elseif (a2(m)==o && b2(m)==p)
                salidaSPtx(o,p)=P;
                q = q-1;
                break;
            else
                salidaSPtx(o,p)=salidaModulador(q);
            end
        end
        q = q+1;
        if (q==(length(salidaModulador)+1))
            break;
        end
    end
    if (q==(length(salidaModulador)+1))
        break;
    end
end



% Rellenamos con 0´s hasta que el número de columnas totales sea múltiplo
% de 7
columnas0 = 7*ceil(columnas/7) - columnas;
salidaSPtx = [salidaSPtx zeros(filas,columnas0)];
[filas1,columnas1] = size(salidaSPtx);

a1 = zeros(1,ceil(filas1/6));
k1 = zeros(1,ceil(filas1/6));
b1 = zeros(1,ceil(columnas1/7));
l1 = zeros(1,ceil(columnas1/7));
r1 = 1;

for m=(GP+0):6:(filas1+0-GP)
    for n=0:7:columnas1
```

```
        aa1 = 6+m;
        a1(1,r1) = aa1;

        c1 = 3+m;
        k1(1,r1) = c1;

        bb1 = 1+n;
        b1(1,r1) = bb1;

        d1 = 5+n;
        l1(1,r1) = d1;

        r1 = r1+1;

    end
end

a2 = zeros(1,ceil(filas1/6));
k2 = zeros(1,ceil(filas1/6));
b2 = zeros(1,ceil(columnas1/7));
l2 = zeros(1,ceil(columnas1/7));
r2 = 1;

for m=(GP+0):6:(filas1+0-GP)
    for n=0:7:columnas1

        aa2 = 6+m;
        a2(1,r2) = aa2;

        c2 = 3+m;
        k2(1,r2) = c2;

        bb2 = 5+n;
        b2(1,r2) = bb2;

        d2 = 1+n;
        l2(1,r2) = d2;

        r2 = r2+1;

    end
end


for p=1:columnas1
    for o=(GP+1):(filas-GP)

        for m=1:length(k1)
            if ((k1(m)==o && l1(m)==p))
                salidaSPtx(o,p)=0;
                break;
            elseif (a1(m)==o && b1(m)==p)
                salidaSPtx(o,p)=0;
                break;
            elseif ((k2(m)==o && l2(m)==p))
                salidaSPtx(o,p)=P;
                break;
            elseif (a2(m)==o && b2(m)==p)
                salidaSPtx(o,p)=P;
                break;
            else
                salidaSPtx(o,p)=salidaSPtx(o,p);
            end
        end
    end
end
```

```
end

--------------------------------------------------------------------------------
function [salidaifft] = iffttxMIMO (salidaSP,Nfft)
% Paso del dominio frecuencial al temporal
[filas,columnas,antenas] = size(salidaSP);
salidaifft = zeros(filas,columnas,antenas);
for m=1:antenas
    for n=1:columnas
        salidaifft(:,n,m) = ifft(salidaSP(:,n,m),Nfft);
    end
end
end

--------------------------------------------------------------------------------

function [salidafft] = fftrxMIMO (salidaPCrx,Nfft)
% Paso del dominio temporal al frecuencial
[filas,columnas,antenas] = size(salidaPCrx);
salidafft = zeros(filas,columnas,antenas);
for m=1:antenas
    for n=1:columnas
        salidafft(:,n,m) = fft(salidaPCrx(:,n,m),Nfft);
    end
end
end

--------------------------------------------------------------------------------

function [H] = ecualizadorMatrizEnteraMIMO2Antena1 (salidafft,P,GP)

[filas,columnas] = size(salidafft);
h = zeros(filas-2*GP,columnas);

% Número pilotos por columna
NP = floor(2*(filas-2*GP)/12);

for j=0:7:(columnas-7)

        % 1ª columna
        a = zeros(NP,1);
        posicion1 = 6;
        for i=1:NP
            if ((GP+posicion1)<(filas-GP+1))
                a(i,1)= salidafft(GP+posicion1,j+1);
                posicion1 = posicion1+6;
            else
                break;
            end
        end
        aifft = ifft(a);
        aifft = [aifft ;zeros((filas-NP-2*GP),1)];
        afft = fft(aifft);
        h(:,j+1) = afft;

        %5ª columna
        b = zeros(NP,1);
        posicion5 = 3;
        for k=1:NP
            if ((GP+posicion5)<(filas-GP+1))
                b(k,1)= salidafft(GP+posicion5,j+5);
                posicion5 = posicion5+6;
            else
                break;
            end
        end
```

```
        bifft = ifft(b);
        bifft = [bifft ;zeros((filas-NP-2*GP),1)];
        bfft = fft(bifft);
        h(:,j+5) = bfft;

end

        % Estimación en la dimensión temporal

        % INTERPOLACIONES

        % Interpolación 1ª y 5ª columna

        for n=1:(filas-2*GP)
            for j=0:7:(columnas-7)

            % Interpolación 1ª y 5ª columna

            a1 = h(n,1+j);
            b1 = h(n,5+j);
            x1 = [1 5];%pos. conocidas
            y1real = [real(a1) real(b1)];%valores de dichas posiciones
            z1real = interp1(x1,y1real,[2 3 4]);
            y1imag = [imag(a1) imag(b1)];
            z1imag = interp1(x1,y1imag,[2 3 4]);
            z21 = z1real(1) +1i*z1imag(1);
            z31 = z1real(2) +1i*z1imag(2);
            z41 = z1real(3) +1i*z1imag(3);
            h(n,2+j) = z21;
            h(n,3+j) = z31;
            h(n,4+j) = z41;

            % Interpolación 5ª y 8ª columna

             if (j==(columnas-7))
                break;
            else
            a2 = h(n,5+j);
            b2 = h(n,8+j);
            x2 = [1 4];%pos. conocidas
            y2real = [real(a2) real(b2)];%valores de dichas posiciones
            z2real = interp1(x2,y2real,[2 3]);
            y2imag = [imag(a2) imag(b2)];
            z2imag = interp1(x2,y2imag,[2 3]);
            z62 = z2real(1) +1i*z2imag(1);
            z72 = z2real(2) +1i*z2imag(2);
            h(n,6+j) = z62;
            h(n,7+j) = z72;
            end

            end
        end

        h(:,columnas-1) = h(:,columnas-2);
        h(:,columnas) = h(:,columnas-2);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%

H1 = zeros(size(h));
[filas1 columnas1] = size(H1);
a = 4;
for i=1:(filas1-a)
    H1(i+a,:) = h(i,:);
end
for j=(filas1-a):filas1
    H1((j-filas1+a+1),:) = h(j,:);
end
```

```
h1 = zeros(GP,columnas);

H = [h1;H1;h1];

end

-------------------------------------------------------------------------------

function [H] = ecualizadorMatrizEnteraMIMO2Antena2 (salidafft,P,GP)

[filas,columnas] = size(salidafft);
h = zeros(filas-2*GP,columnas);

% Número pilotos por columna
NP = floor(2*(filas-2*GP)/12);

for j=0:7:(columnas-7)

        % 1ª columna
        a = zeros(NP,1);
        posicion1 = 3;
        for i=1:NP
            if ((GP+posicion1)<(filas-GP+1))
                a(i,1)= salidafft(GP+posicion1,j+1);
                posicion1 = posicion1+6;
            else
                break;
            end
        end
        aifft = ifft(a);
        aifft = [aifft ;zeros((filas-NP-2*GP),1)];
        afft = fft(aifft);
        h(:,j+1) = afft;

        %5ª columna
        b = zeros(NP,1);
        posicion5 = 6;
        for k=1:NP
            if ((GP+posicion5)<(filas-GP+1))
                b(k,1)= salidafft(GP+posicion5,j+5);
                posicion5 = posicion5+6;
            else
                break;
            end
        end
        bifft = ifft(b);
        bifft = [bifft ;zeros((filas-NP-2*GP),1)];
        bfft = fft(bifft);
        h(:,j+5) = bfft;

end

        % Estimación en la dimensión temporal

        % INTERPOLACIONES

        % Interpolación 1ª y 5ª columna

        for n=1:(filas-2*GP)
            for j=0:7:(columnas-7)

            % Interpolación 1ª y 5ª columna

            a1 = h(n,1+j);
            b1 = h(n,5+j);
```

```
            x1 = [1 5];%pos. conocidas
            y1real = [real(a1) real(b1)];%valores de dichas posiciones
            z1real = interp1(x1,y1real,[2 3 4]);
            y1imag = [imag(a1) imag(b1)];
            z1imag = interp1(x1,y1imag,[2 3 4]);
            z21 = z1real(1) +1i*z1imag(1);
            z31 = z1real(2) +1i*z1imag(2);
            z41 = z1real(3) +1i*z1imag(3);
            h(n,2+j) = z21;
            h(n,3+j) = z31;
            h(n,4+j) = z41;

            % Interpolación 5ª y 8ª columna

             if (j==(columnas-7))
                break;
            else
            a2 = h(n,5+j);
            b2 = h(n,8+j);
            x2 = [1 4];%pos. conocidas
            y2real = [real(a2) real(b2)];%valores de dichas posiciones
            z2real = interp1(x2,y2real,[2 3]);
            y2imag = [imag(a2) imag(b2)];
            z2imag = interp1(x2,y2imag,[2 3]);
            z62 = z2real(1) +1i*z2imag(1);
            z72 = z2real(2) +1i*z2imag(2);
            h(n,6+j) = z62;
            h(n,7+j) = z72;
            end

            end
        end

        h(:,columnas-1) = h(:,columnas-2);
        h(:,columnas) = h(:,columnas-2);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%

H1 = zeros(size(h));
[filas1 columnas1] = size(H1);
a = 4;
for i=1:(filas1-a)
    H1(i+a,:) = h(i,:);
end
for j=(filas1-a):filas1
    H1((j-filas1+a+1),:) = h(j,:);
end

h1 = zeros(GP,columnas);
H = [h1;H1;h1];

end

-------------------------------------------------------------------------------

function [salidaPSrx] = conversorPSrxMIMOAntena1 (salidaEcualizador0,GP,Npilotos)

[filas0,columnas0] = size(salidaEcualizador0);
salidaEcualizador = salidaEcualizador0((GP+1):(filas0-GP),:);

[filas,columnas] = size(salidaEcualizador);
salidaPSrx = zeros(1,filas*columnas-Npilotos);

% Definimos la ubicación de las señales piloto
a1 = zeros(1,ceil(filas/6));
k1 = zeros(1,ceil(filas/6));
b1 = zeros(1,ceil(columnas/7));
```

```
l1 = zeros(1,ceil(columnas/7));
p1 = 1;

for i=0:6:(filas-1)
    for j=0:7:(columnas-1)

        aa1 = 3+i;
        a1(1,p1) = aa1;

        c1 = 6+i;
        k1(1,p1) = c1;

        bb1 = 5+j;
        b1(1,p1) = bb1;

        d1 = 1+j;
        l1(1,p1) = d1;

        p1 = p1+1;

    end
end

% Definimos las posiciones no usadas
a2 = zeros(1,ceil(filas/6));
k2 = zeros(1,ceil(filas/6));
b2 = zeros(1,ceil(columnas/7));
l2 = zeros(1,ceil(columnas/7));
p2 = 1;

for i=0:6:(filas-1)
    for j=0:7:(columnas-1)

        aa2 = 3+i;
        a2(1,p2) = aa2;

        c2 = 6+i;
        k2(1,p2) = c2;

        bb2 = 1+j;
        b2(1,p2) = bb2;

        d2 = 5+j;
        l2(1,p2) = d2;

        p2 = p2+1;

    end
end

q=1;
for n=1:columnas
   for o=1:filas
       % Recorremos la matriz con el objetivo de extraer la salida del
       % ecualizador que se trata de un vector lineal
       % Si llegamos a una posición de la matriz correspondiente a una
       % señal piloto no usamos esa posición en la salida del ecualizador, en caso
       % contrario introducimos el símbolo de tráfico del modulador en la
       % posición correspondiente de la salida del ecualizador

       for m=1:length(k1)
            if (k1(m)==o && l1(m)==n)
                q = q-1;
                break;
            elseif (a1(m)==o && b1(m)==n)
                q = q-1;
```

```
                break;
            elseif ((k2(m)==o && l2(m)==n))
                q = q-1;
                break
            elseif (a2(m)==o && b2(m)==n)
                q = q-1;
                break;
            else
                salidaPSrx(q)=salidaEcualizador(o,n);
            end
        end
        q=q+1;
    end
end
end

--------------------------------------------------------------------------------

function [salidaPSrx] = conversorPSrxMIMOAntena2 (salidaEcualizador0,GP,Npilotos)

[filas0,columnas0] = size(salidaEcualizador0);
salidaEcualizador = salidaEcualizador0((GP+1):(filas0-GP),:);

[filas,columnas] = size(salidaEcualizador);
salidaPSrx = zeros(1,filas*columnas-Npilotos);

% Definimos las posiciones no usadas
a1 = zeros(1,ceil(filas/6));
k1 = zeros(1,ceil(filas/6));
b1 = zeros(1,ceil(columnas/7));
l1 = zeros(1,ceil(columnas/7));
p1 = 1;

for i=0:6:(filas-1)
    for j=0:7:(columnas-1)

        aa1 = 3+i;
        a1(1,p1) = aa1;

        c1 = 6+i;
        k1(1,p1) = c1;

        bb1 = 5+j;
        b1(1,p1) = bb1;

        d1 = 1+j;
        l1(1,p1) = d1;

        p1 = p1+1;

    end
end

% Definimos la ubicación de las señales piloto
a2 = zeros(1,ceil(filas/6));
k2 = zeros(1,ceil(filas/6));
b2 = zeros(1,ceil(columnas/7));
l2 = zeros(1,ceil(columnas/7));
p2 = 1;

for i=0:6:(filas-1)
    for j=0:7:(columnas-1)

        aa2 = 3+i;
        a2(1,p2) = aa2;
```

```
        c2 = 6+i;
        k2(1,p2) = c2;

        bb2 = 1+j;
        b2(1,p2) = bb2;

        d2 = 5+j;
        l2(1,p2) = d2;

        p2 = p2+1;

    end
end

q=1;
for n=1:columnas
   for o=1:filas
       % Recorremos la matriz con el objetivo de extraer la salida del
       % ecualizador que se trata de un vector lineal
       % Si llegamos a una posición de la matriz correspondiente a una
       % señal piloto no usamos esa posición en la salida del ecualizador, en caso
       % contrario introducimos el símbolo de tráfico del modulador en la
       % posición correspondiente de la salida del ecualizador

       for m=1:length(k1)
            if (k1(m)==o && l1(m)==n)
                q = q-1;
                break;
            elseif (a1(m)==o && b1(m)==n)
                q = q-1;
                break;
            elseif ((k2(m)==o && l2(m)==n))
                q = q-1;
                break
            elseif (a2(m)==o && b2(m)==n)
                q = q-1;
                break;
            else
                salidaPSrx(q)=salidaEcualizador(o,n);
            end
       end
       q=q+1;
    end
end
end
-------------------------------------------------------------------------------
```

CÓDIGO DEL SISTEMA MIMO 4x4

Archivos necesarios:

- anadirPrefijoCiclico.m
- conversorPSrxMIMOAntena1
- conversorPSrxMIMOAntena2
- conversorPStx.m
- conversorSPrx.m
- conversorSPtxAntena1MaximosPilotos.m
- conversorSPtxAntena1PilotosEstandar.m
- conversorSPtxAntena2MaximosPilotos.m
- conversorSPtxAntena2PilotosEstandar.m
- Demodulador16QAM.m
- Demodulador64QAM.m
- DemoduladorQPSK.m
- ecualizadorMatrizEnteraMIMO2Antena1.m
- ecualizadorMatrizEnteraMIMO2Antena2.m
- ElegirDemodulador.m
- ElegirModulador.m
- eliminarPrefijoCiclico.m
- fftrxMIMO.m
- iffttxMIMO.m
- MIMO2x2.m
- Modulador16QAM.m
- Modulador64QAM.m
- ModuladorQPSK.m

A continuación se muestra el script del código elaborado para el sistema MIMO 4x4 (MIMO4x4.m) con cuatro antenas transmisoras y con cuatro antenas receptoras. Posteriormente se muestran las el resto de funciones, exceptuando aquellas que son las mismas funciones en el sistema SISO y en el sistema MIMO 2x2 y que fueron definidas con anterioridad.

```
clc
clear all
close all

% MIMO: 4 antenas en el tx y 4 en el rx

% Bits de entrada que queremos enviar
%bitsEntrada = round(rand(1,1008*36));
bitsEntrada = round(rand(1,36*360));
% Repartimos los bits de entrada para cada una de las antenas
bitsEntradaAntena1 = bitsEntrada(1:length(bitsEntrada)/4);
bitsEntradaAntena2 = bitsEntrada((length(bitsEntrada)/4+1):length(bitsEntrada)/2);
```

```
bitsEntradaAntena3 =
bitsEntrada((length(bitsEntrada)/2+1):3*length(bitsEntrada)/4);
bitsEntradaAntena4 = bitsEntrada((3*length(bitsEntrada)/4+1):end);

% 1 - TRANSMISOR

% 1.1 - Modulador
% Convertimos los bits de entrada en símbolos

% Elegimos la constelación que queremos mediante la variable tipoConstelacion:
% 1 - Constelación QPSK
% 2 - Constelación 16QAM
% 3 - Constelación 64QAM

tipoConstelacion = 3;

salidaModuladorAntena1 = ElegirModulador(bitsEntradaAntena1,tipoConstelacion);
PotenciaSenalAntena1 = var(salidaModuladorAntena1);
salidaModuladorAntena2 = ElegirModulador(bitsEntradaAntena2,tipoConstelacion);
PotenciaSenalAntena2 = var(salidaModuladorAntena2);
salidaModuladorAntena3 = ElegirModulador(bitsEntradaAntena3,tipoConstelacion);
PotenciaSenalAntena3 = var(salidaModuladorAntena3);
salidaModuladorAntena4 = ElegirModulador(bitsEntradaAntena4,tipoConstelacion);
PotenciaSenalAntena4 = var(salidaModuladorAntena4);
% 1.2 - Conversor Serie Paralelo

Nfft = 128; % Definimos el tamaño de la FFT
P = 1; % Valor de la señal piloto

% Usamos las frecuencias útiles definidas por el estándar
[GP,Npilotos1,salidaSPtxAntena1] =
conversorSPtxAnt1PilotosEstandar(salidaModuladorAntena1,Nfft,P);
[GP1,Npilotos2,salidaSPtxAntena2] =
conversorSPtxAnt2PilotosEstandar(salidaModuladorAntena2,Nfft,P);
[GP2,Npilotos3,salidaSPtxAntena3] =
conversorSPtxAnt3PilotosEstandar(salidaModuladorAntena3,Nfft,P);
[GP3,Npilotos4,salidaSPtxAntena4] =
conversorSPtxAnt4PilotosEstandar(salidaModuladorAntena4,Nfft,P);

% Usamos el número máximo de pilotos posible
% [GP,Npilotos1,salidaSPtxAntena1] =
conversorSPtxAnt1MaximosPilotos(salidaModuladorAntena1,Nfft,P);
% [GP1,Npilotos2,salidaSPtxAntena2] =
conversorSPtxAnt2MaximosPilotos(salidaModuladorAntena2,Nfft,P);
% [GP2,Npilotos3,salidaSPtxAntena3] =
conversorSPtxAnt3MaximosPilotos(salidaModuladorAntena3,Nfft,P);
% [GP3,Npilotos4,salidaSPtxAntena4] =
conversorSPtxAnt4MaximosPilotos(salidaModuladorAntena4,Nfft,P);

[tam1,tam2] = size(salidaSPtxAntena1);
salidaSPtx = zeros(tam1,tam2,4);
salidaSPtx(:,:,1) = salidaSPtxAntena1;
salidaSPtx(:,:,2) = salidaSPtxAntena2;
salidaSPtx(:,:,3) = salidaSPtxAntena3;
salidaSPtx(:,:,4) = salidaSPtxAntena4;

% 1.3 - IFFT
%Pasamos del dominio de la frecuencia al dominio del tiempo
salidaifft = iffttxMIMO(salidaSPtx,Nfft);

% 1.4 - Añadir prefijo cíclico
% La adición del prefijo cíclico consiste en copiar la parte final del
% símbolo al principio del mismo, es decir, el final de la columna del símbolo
% correspondiente, en el principio de la misma
tamPC = Nfft/8; % Tamaño del prefijo cíclico
[filas,columnas,antenas] = size(salidaifft);
```

```
salidaPCtx = zeros(filas+tamPC,columnas);
for m=1:antenas
    for n=1:columnas
        salidaPCtx(:,n,m) = anadirPrefijoCiclico(salidaifft(:,n,m),tamPC);
    end
end

% 1.5 - Conversor Paralelo Serie
salidaPStx1 = conversorPStx(salidaPCtx(:,:,1));
salidaPStx2 = conversorPStx(salidaPCtx(:,:,2));
salidaPStx3 = conversorPStx(salidaPCtx(:,:,3));
salidaPStx4 = conversorPStx(salidaPCtx(:,:,4));

% 2 - CANAL
% Definimos el tipo de canal existente entre el transmisor y el receptor
ni = 0.5;
ni1 =
(ni)*(1/sqrt(2))*(randn(1,length(salidaPStx1))+1i*randn(1,length(salidaPStx1)));
ni2 =
(ni)*(1/sqrt(2))*(randn(1,length(salidaPStx2))+1i*randn(1,length(salidaPStx2)));
ni3 =
(ni)*(1/sqrt(2))*(randn(1,length(salidaPStx1))+1i*randn(1,length(salidaPStx1)));
ni4 =
(ni)*(1/sqrt(2))*(randn(1,length(salidaPStx2))+1i*randn(1,length(salidaPStx2)));
PotenciaRuido1 = var(ni1);
PotenciaRuido2 = var(ni2);
PotenciaRuido3 = var(ni3);
PotenciaRuido4 = var(ni4);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
canal11=[0.909505559686059+0.0570811522831248i
0.0456760830213736+0.111242422458266i -0.247602286506910-0.009283304416878O5i
0.0610869817003726-0.0342405687499177i -0.008029148864771 59+0.0224937836985498i
0.153191551329004+0.130831812483216i];
salidaCanal11 = filter(canal11, 1, salidaPStx1);

canal12=[0.566746906538893+0.630113834041522i 0.0357449752583050-
0.0679886475792722i 0.270114445438092+0.183711628767796i -
0.0778588683297167+0.006747986835070861 -0.0252790329181005-0.0356571054464926i
0.132353771086039+0.157961257698125i]; % dos caminos
salidaCanal12 = filter(canal12, 1, salidaPStx2);

canal13=[-0.474666676819643+0.358708495851864i 0.0406971799009435-
0.130806647107331i 0.0340745753950306-0.0121605503030910i -
0.008957530344932961+0.0408387125448647i -0.0735779229265508-0.0488901056490232i
0.0931540690409374-0.0856279555002441i];
salidaCanal13 = filter(canal13, 1, salidaPStx3);

canal14=[-0.531443401493595+0.2422317277052881 -
0.145146758621639+0.0783346111385592i -0.116511995739920-0.0220821013769710i
0.0270894999888697-0.0403006766054490i 0.0287187715050430+0.0777286940040754i
0.202029013372210-0.0614940293559058i]; % dos caminos
salidaCanal14 = filter(canal14, 1, salidaPStx4);

salidaCanal1 = salidaCanal11+salidaCanal12+salidaCanal13+salidaCanal14;

PotenciaSenal1 = var(salidaCanal1);
norm1 = PotenciaSenalAntena1/PotenciaSenal1;
salidaCanal1 = salidaCanal1.*sqrt(norm1);
PotenciaSenal1 = var(salidaCanal1);
salidaCanal1ConRuido = salidaCanal1+ni1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
canal21=[-0.531443401493595+0.2422317277052881 -
0.145146758621639+0.0783346111385592i -0.116511995739920-0.0220821013769710i
0.0270894999888697-0.0403006766054490i 0.0287187715050430+0.0777286940040754i
0.202029013372210-0.0614940293559058i];
```

```
salidaCanal21 = filter(canal21, 1, salidaPStx1);

canal22=[0.909505559686059+0.0570811522831248i
0.0456760830213736+0.111242422458266i -0.247602286506910-0.009283304416878O5i
0.0610869817003726-0.0342405687499177i -0.008029148864771S9+0.0224937836985498i
0.153191551329004+0.130831812483216i]; % dos caminos
salidaCanal22 = filter(canal22, 1, salidaPStx2);

canal23=[-0.474666676819643+0.358708495851864i 0.0406971799009435-
0.130806647107331i 0.0340745753950306-0.0121605503030910i -
0.008957530344932960+0.0408387125448647i -0.0735779229265508-0.0488901056490232i
0.0931540690409374-0.0856279555002441i];
salidaCanal23 = filter(canal23, 1, salidaPStx3);

canal24=[0.566746906538893+0.630113834041522i 0.0357449752583050-
0.0679886475792722i 0.270114445438092+0.183711628767796i -
0.0778588683297167+0.006747986835070860i -0.0252790329181005-0.0356571054464926i
0.132353771086039+0.157961257698125i]; % dos caminos
salidaCanal24 = filter(canal24, 1, salidaPStx4);

salidaCanal2 = salidaCanal21+salidaCanal22+salidaCanal23+salidaCanal24;

PotenciaSenal2 = var(salidaCanal2);
norm2 = PotenciaSenalAntena2/PotenciaSenal2;
salidaCanal2 = salidaCanal2.*sqrt(norm2);
PotenciaSenal2 = var(salidaCanal2);
salidaCanal2ConRuido = salidaCanal2+ni2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
canal31=[0.566746906538893+0.630113834041522i 0.0357449752583050-
0.0679886475792722i 0.270114445438092+0.183711628767796i -
0.0778588683297167+0.006747986835070860i -0.0252790329181005-0.0356571054464926i
0.132353771086039+0.157961257698125i];
salidaCanal31 = filter(canal31, 1, salidaPStx1);

canal32=[-0.531443401493595+0.242231727705288i -
0.145146758621639+0.078334611138559 2i -0.116511995739920-0.0220821013769710i
0.027089499988869 7-0.040300676605449 0i 0.028718771505043 0+0.077286940040075 4i
0.202029013372210-0.061494029355905 8i]; % dos caminos
salidaCanal32 = filter(canal32, 1, salidaPStx2);

canal33=[-0.474666676819643+0.358708495851864i 0.0406971799009435-
0.130806647107331i 0.0340745753950306-0.0121605503030910i -
0.008957530344932960+0.0408387125448647i -0.0735779229265508-0.0488901056490232i
0.0931540690409374-0.0856279555002441i];
salidaCanal33 = filter(canal33, 1, salidaPStx3);

canal34=[0.909505559686059+0.0570811522831248i
0.0456760830213736+0.111242422458266i -0.247602286506910-0.009283304416878O5i
0.0610869817003726-0.0342405687499177i -0.008029148864771S9+0.0224937836985498i
0.153191551329004+0.130831812483216i]; % dos caminos
salidaCanal34 = filter(canal34, 1, salidaPStx4);

salidaCanal3 = salidaCanal31+salidaCanal32+salidaCanal33+salidaCanal34;

PotenciaSenal3 = var(salidaCanal3);
norm3 = PotenciaSenalAntena3/PotenciaSenal3;
salidaCanal3 = salidaCanal3.*sqrt(norm3);
PotenciaSenal3 = var(salidaCanal3);
salidaCanal3ConRuido = salidaCanal3+ni3;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
canal41=[-0.531443401493595+0.242231727705288i -
0.145146758621639+0.078334611138559 2i -0.116511995739920-0.0220821013769710i
0.027089499988869 7-0.040300676605449 0i 0.028718771505043 0+0.077286940040075 4i
0.202029013372210-0.061494029355905 8i];
```

```
salidaCanal41 = filter(canal41, 1, salidaPStx1);

canal42=[0.566746906538893+0.630113834041522i 0.0357449752583050-
0.0679886475792722i 0.270114445438092+0.183711628767796i -
0.0778588683297167+0.00674798683507086i -0.0252790329181005-0.0356571054464926i
0.132353771086039+0.157961257698125i]; % dos caminos
salidaCanal42 = filter(canal42, 1, salidaPStx2);

canal43=[0.909505559686059+0.0570811522831248i
0.0456760830213736+0.111242422458266i -0.247602286506910-0.00928330441687805i
0.0610869817003726-0.0342405687499177i -0.00802914886477159+0.0224937836985498i
0.153191551329004+0.130831812483216i];
salidaCanal43 = filter(canal43, 1, salidaPStx3);

canal44=[-0.474666676819643+0.358708495851864i 0.0406971799009435-
0.130806647107331i 0.0340745753950306-0.0121605503030910i -
0.00895753034493296+0.0408387125448647i -0.0735779229265508-0.0488901056490232i
0.0931540690409374-0.0856279555002441i]; % dos caminos
salidaCanal44 = filter(canal44, 1, salidaPStx4);

salidaCanal4 = salidaCanal41+salidaCanal42+salidaCanal43+salidaCanal44;

PotenciaSenal4 = var(salidaCanal4);
norm4 = PotenciaSenalAntena4/PotenciaSenal4;
salidaCanal4 = salidaCanal4.*sqrt(norm4);
PotenciaSenal4 = var(salidaCanal4);
salidaCanal4ConRuido = salidaCanal4+ni4;

SNR1dB = 10*log10(PotenciaSenal1/PotenciaRuido1);
SNR2dB = 10*log10(PotenciaSenal2/PotenciaRuido2);
SNR3dB = 10*log10(PotenciaSenal3/PotenciaRuido3);
SNR4dB = 10*log10(PotenciaSenal4/PotenciaRuido4);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% 3 - RECEPTOR

% 3.1 - Conversor Serie Paralelo
salidaSPrx1 = conversorSPrx(salidaCanal1ConRuido,Nfft,tamPC);
salidaSPrx2 = conversorSPrx(salidaCanal2ConRuido,Nfft,tamPC);
salidaSPrx3 = conversorSPrx(salidaCanal3ConRuido,Nfft,tamPC);
salidaSPrx4 = conversorSPrx(salidaCanal4ConRuido,Nfft,tamPC);
[filas,columnas] = size(salidaSPrx1);
salidaSPrx = zeros(filas,columnas,4);
salidaSPrx(:,:,1) = salidaSPrx1(:,:);
salidaSPrx(:,:,2) = salidaSPrx2(:,:);
salidaSPrx(:,:,3) = salidaSPrx3(:,:);
salidaSPrx(:,:,4) = salidaSPrx4(:,:);

% 3.2 - Eliminar Prefijo Cíclico
% Eliminamos el comienzo de cada una de los columnas
% El tamaño del prefijo cíclico está definido anteriormente en tamPC
[filas,columnas,antenas] = size(salidaSPrx);
salidaPCrx = zeros(filas-tamPC,columnas,antenas);
for m=1:antenas
    for n=1:columnas
       salidaPCrx(:,n,m) = eliminarPrefijoCiclico(salidaSPrx(:,n,m),tamPC);
    end
end

% 3.1 - FFT
% Pasamos del dominio del tiempo al dominio frecuencial
salidafft = fftrxMIMO(salidaPCrx,Nfft);

% 3.2 - Ecualizador
% Revierte el impacto del canal multiplicando cada subportadora por un número
```

```
% complejo basado en la estimación del canal
% Obtenemos la matriz H que será la encargada de indicarnos la variación de
% amplitud y fase que ha sufrido cada uno de los elementos de los
% diferentes símbolos
H11 = ecualizadorMatrizEnteraMIMO2Antena1 (salidafft(:,:,1),P,GP);
H12 = ecualizadorMatrizEnteraMIMO2Antena2 (salidafft(:,:,1),P,GP);
H13 = ecualizadorMatrizEnteraMIMOAntena3 (salidafft(:,:,1),P,GP);
H14 = ecualizadorMatrizEnteraMIMOAntena4 (salidafft(:,:,1),P,GP);
H21 = ecualizadorMatrizEnteraMIMO2Antena1 (salidafft(:,:,2),P,GP);
H22 = ecualizadorMatrizEnteraMIMO2Antena2 (salidafft(:,:,2),P,GP);
H23 = ecualizadorMatrizEnteraMIMOAntena3 (salidafft(:,:,2),P,GP);
H24 = ecualizadorMatrizEnteraMIMOAntena4 (salidafft(:,:,2),P,GP);
H31 = ecualizadorMatrizEnteraMIMO2Antena1 (salidafft(:,:,3),P,GP);
H32 = ecualizadorMatrizEnteraMIMO2Antena2 (salidafft(:,:,3),P,GP);
H33 = ecualizadorMatrizEnteraMIMOAntena3 (salidafft(:,:,3),P,GP);
H34 = ecualizadorMatrizEnteraMIMOAntena4 (salidafft(:,:,3),P,GP);
H41 = ecualizadorMatrizEnteraMIMO2Antena1 (salidafft(:,:,4),P,GP);
H42 = ecualizadorMatrizEnteraMIMO2Antena2 (salidafft(:,:,4),P,GP);
H43 = ecualizadorMatrizEnteraMIMOAntena3 (salidafft(:,:,4),P,GP);
H44 = ecualizadorMatrizEnteraMIMOAntena4 (salidafft(:,:,4),P,GP);

[filasH,columnasH] = size(H11);

salidaEcualizador = zeros(filasH,columnasH,2);

for i=1:filasH
    for j=1:columnasH
        H = [H11(i,j) H12(i,j) H13(i,j) H14(i,j); H21(i,j) H22(i,j) H23(i,j)
H24(i,j);H31(i,j) H32(i,j) H33(i,j) H34(i,j);H41(i,j) H42(i,j) H43(i,j) H44(i,j)];
        x = [salidafft(i,j,1) ; salidafft(i,j,2);salidafft(i,j,3) ;
salidafft(i,j,4)];
        z = pinv(H)*x;
        salidaEcualizador(i,j,1) = z(1,1);
        salidaEcualizador(i,j,2) = z(2,1);
        salidaEcualizador(i,j,3) = z(3,1);
        salidaEcualizador(i,j,4) = z(4,1);
    end
end

% Conversor Paralelo Serie
salidaPSrxAntena1 = conversorPSrxMIMO4antenas(salidaEcualizador(:,:,1),GP);
salidaPSrxAntena2 = conversorPSrxMIMO4antenas(salidaEcualizador(:,:,2),GP);
salidaPSrxAntena3 = conversorPSrxMIMO4antenas(salidaEcualizador(:,:,3),GP);
salidaPSrxAntena4 = conversorPSrxMIMO4antenas(salidaEcualizador(:,:,4),GP);

% Demodulador
% Convierte los símbolos en una cadena de bits
% tipoConstelacion ya esta definido al comienzo de la tranmisión
salidaDemodulador1 =  ElegirDemodulador(salidaPSrxAntena1,tipoConstelacion);
salidaDemoduladorAntena1 = salidaDemodulador1(1:length(bitsEntradaAntena1));
salidaDemodulador2 =  ElegirDemodulador(salidaPSrxAntena2,tipoConstelacion);
salidaDemoduladorAntena2 = salidaDemodulador2(1:length(bitsEntradaAntena2));
salidaDemodulador3 =  ElegirDemodulador(salidaPSrxAntena3,tipoConstelacion);
salidaDemoduladorAntena3 = salidaDemodulador3(1:length(bitsEntradaAntena3));
salidaDemodulador4 =  ElegirDemodulador(salidaPSrxAntena4,tipoConstelacion);
salidaDemoduladorAntena4 = salidaDemodulador4(1:length(bitsEntradaAntena4));

% Calculamos los porcentajes de fallos
fallos1 = 0;
for i=1:length(bitsEntradaAntena1)
    b1 = (abs(salidaDemoduladorAntena1(i)-bitsEntradaAntena1(i)));
    fallos1 = fallos1+b1;
end
PorcentajeFallosAntena1 = (fallos1*100)/length(bitsEntradaAntena1);

fallos2 = 0;
```

```
for i=1:length(bitsEntradaAntena2)
    b2 = (abs(salidaDemoduladorAntena2(i)-bitsEntradaAntena2(i)));
    fallos2 = fallos2+b2;
end
PorcentajeFallosAntena2 = (fallos2*100)/length(bitsEntradaAntena2);

fallos3 = 0;
for i=1:length(bitsEntradaAntena3)
    b3 = (abs(salidaDemoduladorAntena3(i)-bitsEntradaAntena3(i)));
    fallos3 = fallos3+b3;
end
PorcentajeFallosAntena3 = (fallos3*100)/length(bitsEntradaAntena3);

fallos4 = 0;
for i=1:length(bitsEntradaAntena4)
    b4 = (abs(salidaDemoduladorAntena4(i)-bitsEntradaAntena4(i)));
    fallos4 = fallos4+b4;
end
PorcentajeFallosAntena4 = (fallos4*100)/length(bitsEntradaAntena4);

--------------------------------------------------------------------------------

function [GP,Npilotos,salidaSPtx] = conversorSPtxAnt1PilotosEstandar
(salidaModulador,Nfft,P)

% Convierte los símbolos que salen en serie del modulador en una matriz,
% cuyas filas se corresponden con las diferentes frecuencias portadoras y
% cuyas columnas se corresponden con los slots de tiempo.
% El número de filas vendrá definido por el valor de Nfft y existirán unas
% frecuencias de guarda.
% Se insertaran las señales piloto (de valor P) que servirán para la
% estimación del canal en el ecualizador
% Cada símbolo es una columna completa definida en un slot de tiempo
% concreto

n = length(salidaModulador);
Npilotos = floor(8*n/(12*14));
Nceros = 2*Npilotos;
filas = Nfft;
switch (Nfft)
    case 128
        GP = (Nfft-72)/2;
    case 256
        GP = (Nfft-180)/2;
    case 512
        GP = (Nfft-300)/2;
    case 1024
        GP = (Nfft-600)/2;
    case 1536
        GP = (Nfft-900)/2;
    case 2048
        GP = (Nfft-1200)/2;
end
columnas = ceil((n+Npilotos+Nceros)/(filas-2*GP))+1;
salidaSPtx = zeros(filas,columnas);

% Definimos la ubicación de las señales piloto
a1 = zeros(1,ceil(filas/6));
k1 = zeros(1,ceil(filas/6));
b1 = zeros(1,ceil(columnas/7));
l1 = zeros(1,ceil(columnas/7));
p1 = 1;

for i=(GP+0):6:(filas+0-GP)
    for j=0:7:columnas
```

```
        aa1 = 3+i;
        a1(1,p1) = aa1;

        c1 = 6+i;
        k1(1,p1) = c1;

        bb1 = 5+j;
        b1(1,p1) = bb1;

        d1 = 1+j;
        l1(1,p1) = d1;

        p1 = p1+1;

    end
end

% Definimos las posiciones no usadas
a2 = zeros(1,ceil(filas/6));
k2 = zeros(1,ceil(filas/6));
b2 = zeros(1,ceil(columnas/7));
l2 = zeros(1,ceil(columnas/7));
p2 = 1;

for i=(GP+0):6:(filas+0-GP)
    for j=0:7:columnas

        aa2 = 3+i;
        a2(1,p2) = aa2;

        c2 = 6+i;
        k2(1,p2) = c2;

        bb2 = 1+j;
        b2(1,p2) = bb2;

        d2 = 5+j;
        l2(1,p2) = d2;

        p2 = p2+1;

    end
end

% Definimos las nuevas posiciones no usadas (Antena3)

a3 = zeros(1,ceil(filas/6));
k3 = zeros(1,ceil(filas/6));
b3 = zeros(1,ceil(columnas/14));
l3 = zeros(1,ceil(columnas/14));
r3 = 1;

for i=(GP+0):6:(filas+0-GP)
    for j=0:14:columnas

        aa3 = 3+i;
        a3(1,r3) = aa3;

        c3 = 6+i;
        k3(1,r3) = c3;

        bb3 = 2+j;
        b3(1,r3) = bb3;

        d3 = 9+j;
        l3(1,r3) = d3;
```

```
        r3 = r3+1;
    end
end

a33 = zeros(length(a2));
a33(1:length(a3)) = a3;
k33 = zeros(length(k2));
k33(1:length(k3)) = k3;
b33 = zeros(length(b2));
b33(1:length(b3)) = b3;
l33 = zeros(length(l2));
l33(1:length(l3)) = l3;

% Definimos las nuevas posiciones no usadas (Antena4)

a4 = zeros(1,ceil(filas/6));
k4 = zeros(1,ceil(filas/6));
b4 = zeros(1,ceil(columnas/14));
l4 = zeros(1,ceil(columnas/14));
r4 = 1;

for i=(GP+0):6:(filas+0-GP)
    for j=0:14:columnas

        aa4 = 6+i;
        a4(1,r4) = aa4;

        c4 = 3+i;
        k4(1,r4) = c4;

        bb4 = 2+j;
        b4(1,r4) = bb4;

        d4 = 9+j;
        l4(1,r4) = d4;

        r4 = r4+1;
    end
end

a44 = zeros(length(a2));
a44(1:length(a4)) = a4;
k44 = zeros(length(k2));
k44(1:length(k4)) = k4;
b44 = zeros(length(b2));
b44(1:length(b4)) = b4;
l44 = zeros(length(l2));
l44(1:length(l4)) = l4;

% Relleno de la matriz con el tráfico
% y con las señales piloto de referencia y las posiciones sin usar

q = 1;
for p=1:columnas
    for o=(GP+1):(filas-GP)
        % Recorremos la matriz
        % Si llegamos a una posición de la matriz correspondiente a una
        % señal piloto rellenamos la posición con el valor P, en caso
        % contrario introducimos el símbolo de tráfico del modulador que
        % corresponda

        for m=1:length(k1)
            if ((k1(m)==o && l1(m)==p))
                salidaSPtx(o,p)=P;
                q = q-1;
```

```
                break;
            elseif (a1(m)==o && b1(m)==p)
                salidaSPtx(o,p)=P;
                q = q-1;
                break;
            elseif ((k2(m)==o && l2(m)==p))
                salidaSPtx(o,p)=0;
                q = q-1;
                break;
            elseif (a2(m)==o && b2(m)==p)
                salidaSPtx(o,p)=0;
                q = q-1;
            elseif (a33(m)==o && b33(m)==p)
                salidaSPtx(o,p)=0;
                q = q-1;
            elseif ((k33(m)==o && l33(m)==p))
                salidaSPtx(o,p)=0;
                q = q-1;
            elseif (a44(m)==o && b44(m)==p)
                salidaSPtx(o,p)=0;
                q = q-1;
            elseif ((k44(m)==o && l44(m)==p))
                salidaSPtx(o,p)=0;
                q = q-1;
                break;
            else
                salidaSPtx(o,p)=salidaModulador(q);
            end
        end
        q = q+1;
        if (q==(length(salidaModulador)+1))
            break;
        end
    end
    if (q==(length(salidaModulador)+1))
        break;
    end
end

% Rellenamos con 0´s hasta que el número de columnas totales sea múltiplo
% de 14
columnas0 = 14*ceil(columnas/14) - columnas;
salidaSPtx = [salidaSPtx zeros(filas,columnas0)];
[filas1,columnas1] = size(salidaSPtx);

% Definimos la ubicación de las señales piloto
a1 = zeros(1,ceil(filas1/6));
k1 = zeros(1,ceil(filas1/6));
b1 = zeros(1,ceil(columnas1/7));
l1 = zeros(1,ceil(columnas1/7));
p1 = 1;

for i=(GP+0):6:(filas1+0-GP)
    for j=0:7:columnas1

        aa1 = 3+i;
        a1(1,p1) = aa1;

        c1 = 6+i;
        k1(1,p1) = c1;

        bb1 = 5+j;
        b1(1,p1) = bb1;

        d1 = 1+j;
        l1(1,p1) = d1;
```

```
        p1 = p1+1;

    end
end

% Definimos las posiciones no usadas
a2 = zeros(1,ceil(filas1/6));
k2 = zeros(1,ceil(filas1/6));
b2 = zeros(1,ceil(columnas1/7));
l2 = zeros(1,ceil(columnas1/7));
p2 = 1;

for i=(GP+0):6:(filas1+0-GP)
    for j=0:7:columnas1

        aa2 = 3+i;
        a2(1,p2) = aa2;

        c2 = 6+i;
        k2(1,p2) = c2;

        bb2 = 1+j;
        b2(1,p2) = bb2;

        d2 = 5+j;
        l2(1,p2) = d2;

        p2 = p2+1;

    end
end

% Definimos las nuevas posiciones no usadas (Antena3)

a3 = zeros(1,ceil(filas1/6));
k3 = zeros(1,ceil(filas1/6));
b3 = zeros(1,ceil(columnas1/14));
l3 = zeros(1,ceil(columnas1/14));
r3 = 1;

for i=(GP+0):6:(filas1+0-GP)
    for j=0:14:columnas1

        aa3 = 3+i;
        a3(1,r3) = aa3;

        c3 = 6+i;
        k3(1,r3) = c3;

        bb3 = 2+j;
        b3(1,r3) = bb3;

        d3 = 9+j;
        l3(1,r3) = d3;

        r3 = r3+1;
    end
end

a33 = zeros(length(a2));
a33(1:length(a3)) = a3;
k33 = zeros(length(k2));
k33(1:length(k3)) = k3;
b33 = zeros(length(b2));
b33(1:length(b3)) = b3;
```

```
l33 = zeros(length(l2));
l33(1:length(l3)) = l3;

% Definimos las nuevas posiciones no usadas (Antena4)

a4 = zeros(1,ceil(filas1/6));
k4 = zeros(1,ceil(filas1/6));
b4 = zeros(1,ceil(columnas1/14));
l4 = zeros(1,ceil(columnas1/14));
r4 = 1;

for i=(GP+0):6:(filas1+0-GP)
    for j=0:14:columnas1

        aa4 = 6+i;
        a4(1,r4) = aa4;

        c4 = 3+i;
        k4(1,r4) = c4;

        bb4 = 2+j;
        b4(1,r4) = bb4;

        d4 = 9+j;
        l4(1,r4) = d4;

        r4 = r4+1;
    end
end

a44 = zeros(length(a2));
a44(1:length(a4)) = a4;
k44 = zeros(length(k2));
k44(1:length(k4)) = k4;
b44 = zeros(length(b2));
b44(1:length(b4)) = b4;
l44 = zeros(length(l2));
l44(1:length(l4)) = l4;

for p=1:columnas1
    for o=(GP+1):(filas1-GP)

        for m=1:length(k1)
            if ((k1(m)==o && l1(m)==p))
                salidaSPtx(o,p)=P;
                break;
            elseif (a1(m)==o && b1(m)==p)
                salidaSPtx(o,p)=P;
                break;
            elseif ((k2(m)==o && l2(m)==p))
                salidaSPtx(o,p)=0;
                break;
            elseif (a2(m)==o && b2(m)==p)
                salidaSPtx(o,p)=0;
                break;
            elseif (a33(m)==o && b33(m)==p)
                salidaSPtx(o,p)=0;
                break;
            elseif (k33(m)==o && l33(m)==p)
                salidaSPtx(o,p)=0;
                break;
            elseif (a44(m)==o && b44(m)==p)
                salidaSPtx(o,p)=0;
                break;
            elseif (k44(m)==o && l44(m)==p)
                salidaSPtx(o,p)=0;
```

```
                break;
            else
                salidaSPtx(o,p)=salidaSPtx(o,p);
            end
        end
    end
end
end

--------------------------------------------------------------------------------

function [GP,Npilotos,salidaSPtx] = conversorSPtxAnt2PilotosEstandar
(salidaModulador,Nfft,P)

% Convierte los símbolos que salen en serie del modulador en una matriz,
% cuyas filas se corresponden con las diferentes frecuencias portadoras y
% cuyas columnas se corresponden con los slots de tiempo.
% El número de filas vendrá definido por el valor de Nfft y existirán unas
% frecuencias de guarda.
% Se insertaran las señales piloto (de valor P) que servirán para la
% estimación del canal en el ecualizador
% Cada símbolo es una columna completa definida en un slot de tiempo
% concreto

n = length(salidaModulador);
Npilotos = floor(8*n/(12*14));
Nceros = 2*Npilotos;
filas = Nfft;
switch (Nfft)
    case 128
        GP = (Nfft-72)/2;
    case 256
        GP = (Nfft-180)/2;
    case 512
        GP = (Nfft-300)/2;
    case 1024
        GP = (Nfft-600)/2;
    case 1536
        GP = (Nfft-900)/2;
    case 2048
        GP = (Nfft-1200)/2;
end
columnas = ceil((n+Npilotos+Nceros)/(filas-2*GP))+1;
salidaSPtx = zeros(filas,columnas);

% Definimos la ubicación de las señales piloto
a1 = zeros(1,ceil(filas/6));
k1 = zeros(1,ceil(filas/6));
b1 = zeros(1,ceil(columnas/7));
l1 = zeros(1,ceil(columnas/7));
p1 = 1;

for i=(GP+0):6:(filas+0-GP)
    for j=0:7:columnas

        aa1 = 3+i;
        a1(1,p1) = aa1;

        c1 = 6+i;
        k1(1,p1) = c1;

        bb1 = 5+j;
        b1(1,p1) = bb1;

        d1 = 1+j;
        l1(1,p1) = d1;
```

```
        p1 = p1+1;

    end
end

% Definimos las posiciones no usadas
a2 = zeros(1,ceil(filas/6));
k2 = zeros(1,ceil(filas/6));
b2 = zeros(1,ceil(columnas/7));
l2 = zeros(1,ceil(columnas/7));
p2 = 1;

for i=(GP+0):6:(filas+0-GP)
    for j=0:7:columnas

        aa2 = 3+i;
        a2(1,p2) = aa2;

        c2 = 6+i;
        k2(1,p2) = c2;

        bb2 = 1+j;
        b2(1,p2) = bb2;

        d2 = 5+j;
        l2(1,p2) = d2;

        p2 = p2+1;

    end
end

% Definimos las nuevas posiciones no usadas (Antena3)

a3 = zeros(1,ceil(filas/6));
k3 = zeros(1,ceil(filas/6));
b3 = zeros(1,ceil(columnas/14));
l3 = zeros(1,ceil(columnas/14));
r3 = 1;

for i=(GP+0):6:(filas+0-GP)
    for j=0:14:columnas

        aa3 = 3+i;
        a3(1,r3) = aa3;

        c3 = 6+i;
        k3(1,r3) = c3;

        bb3 = 2+j;
        b3(1,r3) = bb3;

        d3 = 9+j;
        l3(1,r3) = d3;

        r3 = r3+1;
    end
end

a33 = zeros(length(a2));
a33(1:length(a3)) = a3;
k33 = zeros(length(k2));
k33(1:length(k3)) = k3;
b33 = zeros(length(b2));
b33(1:length(b3)) = b3;
```

```
l33 = zeros(length(l2));
l33(1:length(l3)) = l3;

% Definimos las nuevas posiciones no usadas (Antena4)

a4 = zeros(1,ceil(filas/6));
k4 = zeros(1,ceil(filas/6));
b4 = zeros(1,ceil(columnas/14));
l4 = zeros(1,ceil(columnas/14));
r4 = 1;

for i=(GP+0):6:(filas+0-GP)
    for j=0:14:columnas

        aa4 = 6+i;
        a4(1,r4) = aa4;

        c4 = 3+i;
        k4(1,r4) = c4;

        bb4 = 2+j;
        b4(1,r4) = bb4;

        d4 = 9+j;
        l4(1,r4) = d4;

        r4 = r4+1;
    end
end

a44 = zeros(length(a2));
a44(1:length(a4)) = a4;
k44 = zeros(length(k2));
k44(1:length(k4)) = k4;
b44 = zeros(length(b2));
b44(1:length(b4)) = b4;
l44 = zeros(length(l2));
l44(1:length(l4)) = l4;

% Relleno de la matriz con el tráfico
% y con las señales piloto de referencia y las posiciones sin usar

q = 1;
for p=1:columnas
    for o=(GP+1):(filas-GP)
        % Recorremos la matriz
        % Si llegamos a una posición de la matriz correspondiente a una
        % señal piloto rellenamos la posición con el valor P, en caso
        % contrario introducimos el símbolo de tráfico del modulador que
        % corresponda

        for m=1:length(k1)
            if ((k1(m)==o && l1(m)==p))
                salidaSPtx(o,p)=0;
                q = q-1;
                break;
            elseif (a1(m)==o && b1(m)==p)
                salidaSPtx(o,p)=0;
                q = q-1;
                break;
            elseif ((k2(m)==o && l2(m)==p))
                salidaSPtx(o,p)=P;
                q = q-1;
                break;
            elseif (a2(m)==o && b2(m)==p)
                salidaSPtx(o,p)=P;
```

```
                q = q-1;
            elseif (a33(m)==o && b33(m)==p)
                salidaSPtx(o,p)=0;
                q = q-1;
            elseif ((k33(m)==o && l33(m)==p))
                salidaSPtx(o,p)=0;
                q = q-1;
            elseif (a44(m)==o && b44(m)==p)
                salidaSPtx(o,p)=0;
                q = q-1;
            elseif ((k44(m)==o && l44(m)==p))
                salidaSPtx(o,p)=0;
                q = q-1;
                break;
            else
                salidaSPtx(o,p)=salidaModulador(q);
            end
        end
        q = q+1;
        if (q==(length(salidaModulador)+1))
            break;
        end
    end
    if (q==(length(salidaModulador)+1))
        break;
    end
end

% Rellenamos con 0´s hasta que el número de columnas totales sea múltiplo
% de 14
columnas0 = 14*ceil(columnas/14) - columnas;
salidaSPtx = [salidaSPtx zeros(filas,columnas0)];
[filas1,columnas1] = size(salidaSPtx);

% Definimos la ubicación de las señales piloto
a1 = zeros(1,ceil(filas1/6));
k1 = zeros(1,ceil(filas1/6));
b1 = zeros(1,ceil(columnas1/7));
l1 = zeros(1,ceil(columnas1/7));
p1 = 1;

for i=(GP+0):6:(filas1+0-GP)
    for j=0:7:columnas1

        aa1 = 3+i;
        a1(1,p1) = aa1;

        c1 = 6+i;
        k1(1,p1) = c1;

        bb1 = 5+j;
        b1(1,p1) = bb1;

        d1 = 1+j;
        l1(1,p1) = d1;

        p1 = p1+1;

    end
end

% Definimos las posiciones no usadas
a2 = zeros(1,ceil(filas1/6));
k2 = zeros(1,ceil(filas1/6));
b2 = zeros(1,ceil(columnas1/7));
l2 = zeros(1,ceil(columnas1/7));
```

```
p2 = 1;

for i=(GP+0):6:(filas1+0-GP)
    for j=0:7:columnas1

        aa2 = 3+i;
        a2(1,p2) = aa2;

        c2 = 6+i;
        k2(1,p2) = c2;

        bb2 = 1+j;
        b2(1,p2) = bb2;

        d2 = 5+j;
        l2(1,p2) = d2;

        p2 = p2+1;

    end
end

% Definimos las nuevas posiciones no usadas (Antena3)

a3 = zeros(1,ceil(filas1/6));
k3 = zeros(1,ceil(filas1/6));
b3 = zeros(1,ceil(columnas1/14));
l3 = zeros(1,ceil(columnas1/14));
r3 = 1;

for i=(GP+0):6:(filas1+0-GP)
    for j=0:14:columnas1

        aa3 = 3+i;
        a3(1,r3) = aa3;

        c3 = 6+i;
        k3(1,r3) = c3;

        bb3 = 2+j;
        b3(1,r3) = bb3;

        d3 = 9+j;
        l3(1,r3) = d3;

        r3 = r3+1;
    end
end

a33 = zeros(length(a2));
a33(1:length(a3)) = a3;
k33 = zeros(length(k2));
k33(1:length(k3)) = k3;
b33 = zeros(length(b2));
b33(1:length(b3)) = b3;
l33 = zeros(length(l2));
l33(1:length(l3)) = l3;

% Definimos las nuevas posiciones no usadas (Antena4)

a4 = zeros(1,ceil(filas1/6));
k4 = zeros(1,ceil(filas1/6));
b4 = zeros(1,ceil(columnas1/14));
l4 = zeros(1,ceil(columnas1/14));
r4 = 1;
```

```
for i=(GP+0):6:(filas1+0-GP)
    for j=0:14:columnas1

        aa4 = 6+i;
        a4(1,r4) = aa4;

        c4 = 3+i;
        k4(1,r4) = c4;

        bb4 = 2+j;
        b4(1,r4) = bb4;

        d4 = 9+j;
        l4(1,r4) = d4;

        r4 = r4+1;
    end
end

a44 = zeros(length(a2));
a44(1:length(a4)) = a4;
k44 = zeros(length(k2));
k44(1:length(k4)) = k4;
b44 = zeros(length(b2));
b44(1:length(b4)) = b4;
l44 = zeros(length(l2));
l44(1:length(l4)) = l4;

for p=1:columnas1
    for o=(GP+1):(filas1-GP)

        for m=1:length(k1)
            if ((k1(m)==o && l1(m)==p))
                salidaSPtx(o,p)=0;
                break;
            elseif (a1(m)==o && b1(m)==p)
                salidaSPtx(o,p)=0;
                break;
            elseif ((k2(m)==o && l2(m)==p))
                salidaSPtx(o,p)=P;
                break;
            elseif (a2(m)==o && b2(m)==p)
                salidaSPtx(o,p)=P;
                break;
            elseif (a33(m)==o && b33(m)==p)
                salidaSPtx(o,p)=0;
                break;
            elseif (k33(m)==o && l33(m)==p)
                salidaSPtx(o,p)=0;
                break;
            elseif (a44(m)==o && b44(m)==p)
                salidaSPtx(o,p)=0;
                break;
            elseif (k44(m)==o && l44(m)==p)
                salidaSPtx(o,p)=0;
                break;
            else
                salidaSPtx(o,p)=salidaSPtx(o,p);
            end
        end
    end
end
end

--------------------------------------------------------------------------------
```

```
function [GP,Npilotos,salidaSPtx] = conversorSPtxAnt3PilotosEstandar
(salidaModulador,Nfft,P)

% Convierte los símbolos que salen en serie del modulador en una matriz,
% cuyas filas se corresponden con las diferentes frecuencias portadoras y
% cuyas columnas se corresponden con los slots de tiempo.
% El número de filas vendrá definido por el valor de Nfft y existirán unas
% frecuencias de guarda.
% Se insertaran las señales piloto (de valor P) que servirán para la
% estimación del canal en el ecualizador
% Cada símbolo es una columna completa definida en un slot de tiempo
% concreto

n = length(salidaModulador);
Npilotos = floor(4*n/(12*14));
Nceros = 5*Npilotos;
filas = Nfft;
switch (Nfft)
    case 128
        GP = (Nfft-72)/2;
    case 256
        GP = (Nfft-180)/2;
    case 512
        GP = (Nfft-300)/2;
    case 1024
        GP = (Nfft-600)/2;
    case 1536
        GP = (Nfft-900)/2;
    case 2048
        GP = (Nfft-1200)/2;
end
columnas = ceil((n+Npilotos+Nceros)/(filas-2*GP))+1;
salidaSPtx = zeros(filas,columnas);

% Definimos la ubicación de las señales piloto
a1 = zeros(1,ceil(filas/6));
k1 = zeros(1,ceil(filas/6));
b1 = zeros(1,ceil(columnas/7));
l1 = zeros(1,ceil(columnas/7));
p1 = 1;

for i=(GP+0):6:(filas+0-GP)
    for j=0:7:columnas

        aa1 = 3+i;
        a1(1,p1) = aa1;

        c1 = 6+i;
        k1(1,p1) = c1;

        bb1 = 5+j;
        b1(1,p1) = bb1;

        d1 = 1+j;
        l1(1,p1) = d1;

        p1 = p1+1;

    end
end

% Definimos las posiciones no usadas
a2 = zeros(1,ceil(filas/6));
k2 = zeros(1,ceil(filas/6));
b2 = zeros(1,ceil(columnas/7));
l2 = zeros(1,ceil(columnas/7));
```

```
p2 = 1;

for i=(GP+0):6:(filas+0-GP)
    for j=0:7:columnas

        aa2 = 3+i;
        a2(1,p2) = aa2;

        c2 = 6+i;
        k2(1,p2) = c2;

        bb2 = 1+j;
        b2(1,p2) = bb2;

        d2 = 5+j;
        l2(1,p2) = d2;

        p2 = p2+1;

    end
end

% Definimos las nuevas posiciones no usadas (Antena3)

a3 = zeros(1,ceil(filas/6));
k3 = zeros(1,ceil(filas/6));
b3 = zeros(1,ceil(columnas/14));
l3 = zeros(1,ceil(columnas/14));
r3 = 1;

for i=(GP+0):6:(filas+0-GP)
    for j=0:14:columnas

        aa3 = 3+i;
        a3(1,r3) = aa3;

        c3 = 6+i;
        k3(1,r3) = c3;

        bb3 = 2+j;
        b3(1,r3) = bb3;

        d3 = 9+j;
        l3(1,r3) = d3;

        r3 = r3+1;
    end
end

a33 = zeros(length(a2));
a33(1:length(a3)) = a3;
k33 = zeros(length(k2));
k33(1:length(k3)) = k3;
b33 = zeros(length(b2));
b33(1:length(b3)) = b3;
l33 = zeros(length(l2));
l33(1:length(l3)) = l3;

% Definimos las nuevas posiciones no usadas (Antena4)

a4 = zeros(1,ceil(filas/6));
k4 = zeros(1,ceil(filas/6));
b4 = zeros(1,ceil(columnas/14));
l4 = zeros(1,ceil(columnas/14));
r4 = 1;
```

```
for i=(GP+0):6:(filas+0-GP)
    for j=0:14:columnas

        aa4 = 6+i;
        a4(1,r4) = aa4;

        c4 = 3+i;
        k4(1,r4) = c4;

        bb4 = 2+j;
        b4(1,r4) = bb4;

        d4 = 9+j;
        l4(1,r4) = d4;

        r4 = r4+1;
    end
end

a44 = zeros(length(a2));
a44(1:length(a4)) = a4;
k44 = zeros(length(k2));
k44(1:length(k4)) = k4;
b44 = zeros(length(b2));
b44(1:length(b4)) = b4;
l44 = zeros(length(l2));
l44(1:length(l4)) = l4;

% Relleno de la matriz con el tráfico
% y con las señales piloto de referencia y las posiciones sin usar

q = 1;
for p=1:columnas
    for o=(GP+1):(filas-GP)
        % Recorremos la matriz
        % Si llegamos a una posición de la matriz correspondiente a una
        % señal piloto rellenamos la posición con el valor P, en caso
        % contrario introducimos el símbolo de tráfico del modulador que
        % corresponda

        for m=1:length(k1)
            if ((k1(m)==o && l1(m)==p))
                salidaSPtx(o,p)=0;
                q = q-1;
                break;
            elseif (a1(m)==o && b1(m)==p)
                salidaSPtx(o,p)=0;
                q = q-1;
                break;
            elseif ((k2(m)==o && l2(m)==p))
                salidaSPtx(o,p)=0;
                q = q-1;
                break;
            elseif (a2(m)==o && b2(m)==p)
                salidaSPtx(o,p)=0;
                q = q-1;
            elseif (a33(m)==o && b33(m)==p)
                salidaSPtx(o,p)=0;
                q = q-1;
            elseif ((k33(m)==o && l33(m)==p))
                salidaSPtx(o,p)=0;
                q = q-1;
            elseif (a44(m)==o && b44(m)==p)
                salidaSPtx(o,p)=P;
                q = q-1;
            elseif ((k44(m)==o && l44(m)==p))
```

```
                salidaSPtx(o,p)=P;
                q = q-1;
                break;
            else
                salidaSPtx(o,p)=salidaModulador(q);
            end
        end
        q = q+1;
        if (q==(length(salidaModulador)+1))
            break;
        end
    end
    if (q==(length(salidaModulador)+1))
        break;
    end
end

% Rellenamos con 0´s hasta que el número de columnas totales sea múltiplo
% de 14
columnas0 = 14*ceil(columnas/14) - columnas;
salidaSPtx = [salidaSPtx zeros(filas,columnas0)];
[filas1,columnas1] = size(salidaSPtx);

% Definimos la ubicación de las señales piloto
a1 = zeros(1,ceil(filas1/6));
k1 = zeros(1,ceil(filas1/6));
b1 = zeros(1,ceil(columnas1/7));
l1 = zeros(1,ceil(columnas1/7));
p1 = 1;

for i=(GP+0):6:(filas1+0-GP)
    for j=0:7:columnas1

        aa1 = 3+i;
        a1(1,p1) = aa1;

        c1 = 6+i;
        k1(1,p1) = c1;

        bb1 = 5+j;
        b1(1,p1) = bb1;

        d1 = 1+j;
        l1(1,p1) = d1;

        p1 = p1+1;

    end
end

% Definimos las posiciones no usadas
a2 = zeros(1,ceil(filas1/6));
k2 = zeros(1,ceil(filas1/6));
b2 = zeros(1,ceil(columnas1/7));
l2 = zeros(1,ceil(columnas1/7));
p2 = 1;

for i=(GP+0):6:(filas1+0-GP)
    for j=0:7:columnas1

        aa2 = 3+i;
        a2(1,p2) = aa2;

        c2 = 6+i;
        k2(1,p2) = c2;
```

```
        bb2 = 1+j;
        b2(1,p2) = bb2;

        d2 = 5+j;
        l2(1,p2) = d2;

        p2 = p2+1;

    end
end

% Definimos las nuevas posiciones no usadas (Antena3)

a3 = zeros(1,ceil(filas1/6));
k3 = zeros(1,ceil(filas1/6));
b3 = zeros(1,ceil(columnas1/14));
l3 = zeros(1,ceil(columnas1/14));
r3 = 1;

for i=(GP+0):6:(filas1+0-GP)
    for j=0:14:columnas1

        aa3 = 3+i;
        a3(1,r3) = aa3;

        c3 = 6+i;
        k3(1,r3) = c3;

        bb3 = 2+j;
        b3(1,r3) = bb3;

        d3 = 9+j;
        l3(1,r3) = d3;

        r3 = r3+1;
    end
end

a33 = zeros(length(a2));
a33(1:length(a3)) = a3;
k33 = zeros(length(k2));
k33(1:length(k3)) = k3;
b33 = zeros(length(b2));
b33(1:length(b3)) = b3;
l33 = zeros(length(l2));
l33(1:length(l3)) = l3;

% Definimos las nuevas posiciones no usadas (Antena4)

a4 = zeros(1,ceil(filas1/6));
k4 = zeros(1,ceil(filas1/6));
b4 = zeros(1,ceil(columnas1/14));
l4 = zeros(1,ceil(columnas1/14));
r4 = 1;

for i=(GP+0):6:(filas1+0-GP)
    for j=0:14:columnas1

        aa4 = 6+i;
        a4(1,r4) = aa4;

        c4 = 3+i;
        k4(1,r4) = c4;

        bb4 = 2+j;
        b4(1,r4) = bb4;
```

```
        d4 = 9+j;
        l4(1,r4) = d4;

        r4 = r4+1;
    end
end

a44 = zeros(length(a2));
a44(1:length(a4)) = a4;
k44 = zeros(length(k2));
k44(1:length(k4)) = k4;
b44 = zeros(length(b2));
b44(1:length(b4)) = b4;
l44 = zeros(length(l2));
l44(1:length(l4)) = l4;

for p=1:columnas1
    for o=(GP+1):(filas1-GP)

        for m=1:length(k1)
            if ((k1(m)==o && l1(m)==p))
                salidaSPtx(o,p)=0;
                break;
            elseif (a1(m)==o && b1(m)==p)
                salidaSPtx(o,p)=0;
                break;
            elseif ((k2(m)==o && l2(m)==p))
                salidaSPtx(o,p)=0;
                break;
            elseif (a2(m)==o && b2(m)==p)
                salidaSPtx(o,p)=0;
                break;
            elseif (a33(m)==o && b33(m)==p)
                salidaSPtx(o,p)=0;
                break;
            elseif (k33(m)==o && l33(m)==p)
                salidaSPtx(o,p)=0;
                break;
            elseif (a44(m)==o && b44(m)==p)
                salidaSPtx(o,p)=P;
                break;
            elseif (k44(m)==o && l44(m)==p)
                salidaSPtx(o,p)=P;
                break;
            else
                salidaSPtx(o,p)=salidaSPtx(o,p);
            end
        end
    end
end
end

--------------------------------------------------------------------------------
function [GP,Npilotos,salidaSPtx] = conversorSPtxAnt4PilotosEstandar
(salidaModulador,Nfft,P)

% Convierte los símbolos que salen en serie del modulador en una matriz,
% cuyas filas se corresponden con las diferentes frecuencias portadoras y
% cuyas columnas se corresponden con los slots de tiempo.
% El número de filas vendrá definido por el valor de Nfft y existirán unas
% frecuencias de guarda.
% Se insertaran las señales piloto (de valor P) que servirán para la
% estimación del canal en el ecualizador
% Cada símbolo es una columna completa definida en un slot de tiempo
% concreto
```

```
n = length(salidaModulador);
Npilotos = floor(4*n/(12*14));
Nceros = 5*Npilotos;
filas = Nfft;
switch (Nfft)
    case 128
        GP = (Nfft-72)/2;
    case 256
        GP = (Nfft-180)/2;
    case 512
        GP = (Nfft-300)/2;
    case 1024
        GP = (Nfft-600)/2;
    case 1536
        GP = (Nfft-900)/2;
    case 2048
        GP = (Nfft-1200)/2;
end
columnas = ceil((n+Npilotos+Nceros)/(filas-2*GP))+1;
salidaSPtx = zeros(filas,columnas);

% Definimos la ubicación de las señales piloto
a1 = zeros(1,ceil(filas/6));
k1 = zeros(1,ceil(filas/6));
b1 = zeros(1,ceil(columnas/7));
l1 = zeros(1,ceil(columnas/7));
p1 = 1;

for i=(GP+0):6:(filas+0-GP)
    for j=0:7:columnas

        aa1 = 3+i;
        a1(1,p1) = aa1;

        c1 = 6+i;
        k1(1,p1) = c1;

        bb1 = 5+j;
        b1(1,p1) = bb1;

        d1 = 1+j;
        l1(1,p1) = d1;

        p1 = p1+1;

    end
end

% Definimos las posiciones no usadas
a2 = zeros(1,ceil(filas/6));
k2 = zeros(1,ceil(filas/6));
b2 = zeros(1,ceil(columnas/7));
l2 = zeros(1,ceil(columnas/7));
p2 = 1;

for i=(GP+0):6:(filas+0-GP)
    for j=0:7:columnas

        aa2 = 3+i;
        a2(1,p2) = aa2;

        c2 = 6+i;
        k2(1,p2) = c2;

        bb2 = 1+j;
```

```
        b2(1,p2) = bb2;

        d2 = 5+j;
        l2(1,p2) = d2;

        p2 = p2+1;

    end
end

% Definimos las nuevas posiciones no usadas (Antena3)

a3 = zeros(1,ceil(filas/6));
k3 = zeros(1,ceil(filas/6));
b3 = zeros(1,ceil(columnas/14));
l3 = zeros(1,ceil(columnas/14));
r3 = 1;

for i=(GP+0):6:(filas+0-GP)
    for j=0:14:columnas

        aa3 = 3+i;
        a3(1,r3) = aa3;

        c3 = 6+i;
        k3(1,r3) = c3;

        bb3 = 2+j;
        b3(1,r3) = bb3;

        d3 = 9+j;
        l3(1,r3) = d3;

        r3 = r3+1;
    end
end

a33 = zeros(length(a2));
a33(1:length(a3)) = a3;
k33 = zeros(length(k2));
k33(1:length(k3)) = k3;
b33 = zeros(length(b2));
b33(1:length(b3)) = b3;
l33 = zeros(length(l2));
l33(1:length(l3)) = l3;

% Definimos las nuevas posiciones no usadas (Antena4)

a4 = zeros(1,ceil(filas/6));
k4 = zeros(1,ceil(filas/6));
b4 = zeros(1,ceil(columnas/14));
l4 = zeros(1,ceil(columnas/14));
r4 = 1;

for i=(GP+0):6:(filas+0-GP)
    for j=0:14:columnas

        aa4 = 6+i;
        a4(1,r4) = aa4;

        c4 = 3+i;
        k4(1,r4) = c4;

        bb4 = 2+j;
        b4(1,r4) = bb4;
```

```
        d4 = 9+j;
        l4(1,r4) = d4;

        r4 = r4+1;
    end
end

a44 = zeros(length(a2));
a44(1:length(a4)) = a4;
k44 = zeros(length(k2));
k44(1:length(k4)) = k4;
b44 = zeros(length(b2));
b44(1:length(b4)) = b4;
l44 = zeros(length(l2));
l44(1:length(l4)) = l4;

% Relleno de la matriz con el tráfico
% y con las señales piloto de referencia y las posiciones sin usar

q = 1;
for p=1:columnas
    for o=(GP+1):(filas-GP)
        % Recorremos la matriz
        % Si llegamos a una posición de la matriz correspondiente a una
        % señal piloto rellenamos la posición con el valor P, en caso
        % contrario introducimos el símbolo de tráfico del modulador que
        % corresponda

        for m=1:length(k1)
            if ((k1(m)==o && l1(m)==p))
                salidaSPtx(o,p)=0;
                q = q-1;
                break;
            elseif (a1(m)==o && b1(m)==p)
                salidaSPtx(o,p)=0;
                q = q-1;
                break;
            elseif ((k2(m)==o && l2(m)==p))
                salidaSPtx(o,p)=0;
                q = q-1;
                break;
            elseif (a2(m)==o && b2(m)==p)
                salidaSPtx(o,p)=0;
                q = q-1;
            elseif (a33(m)==o && b33(m)==p)
                salidaSPtx(o,p)=P;
                q = q-1;
            elseif ((k33(m)==o && l33(m)==p))
                salidaSPtx(o,p)=P;
                q = q-1;
            elseif (a44(m)==o && b44(m)==p)
                salidaSPtx(o,p)=0;
                q = q-1;
            elseif ((k44(m)==o && l44(m)==p))
                salidaSPtx(o,p)=0;
                q = q-1;
                break;
            else
                salidaSPtx(o,p)=salidaModulador(q);
            end
        end
        q = q+1;
        if (q==(length(salidaModulador)+1))
            break;
        end
    end
```

```
    if (q==(length(salidaModulador)+1))
        break;
    end
end

% Rellenamos con 0´s hasta que el número de columnas totales sea múltiplo
% de 14
columnas0 = 14*ceil(columnas/14) - columnas;
salidaSPtx = [salidaSPtx zeros(filas,columnas0)];
[filas1,columnas1] = size(salidaSPtx);

% Definimos la ubicación de las señales piloto
a1 = zeros(1,ceil(filas1/6));
k1 = zeros(1,ceil(filas1/6));
b1 = zeros(1,ceil(columnas1/7));
l1 = zeros(1,ceil(columnas1/7));
p1 = 1;

for i=(GP+0):6:(filas1+0-GP)
    for j=0:7:columnas1

        aa1 = 3+i;
        a1(1,p1) = aa1;

        c1 = 6+i;
        k1(1,p1) = c1;

        bb1 = 5+j;
        b1(1,p1) = bb1;

        d1 = 1+j;
        l1(1,p1) = d1;

        p1 = p1+1;

    end
end

% Definimos las posiciones no usadas
a2 = zeros(1,ceil(filas1/6));
k2 = zeros(1,ceil(filas1/6));
b2 = zeros(1,ceil(columnas1/7));
l2 = zeros(1,ceil(columnas1/7));
p2 = 1;

for i=(GP+0):6:(filas1+0-GP)
    for j=0:7:columnas1

        aa2 = 3+i;
        a2(1,p2) = aa2;

        c2 = 6+i;
        k2(1,p2) = c2;

        bb2 = 1+j;
        b2(1,p2) = bb2;

        d2 = 5+j;
        l2(1,p2) = d2;

        p2 = p2+1;

    end
end

% Definimos las nuevas posiciones no usadas (Antena3)
```

```
a3 = zeros(1,ceil(filas1/6));
k3 = zeros(1,ceil(filas1/6));
b3 = zeros(1,ceil(columnas1/14));
l3 = zeros(1,ceil(columnas1/14));
r3 = 1;

for i=(GP+0):6:(filas1+0-GP)
    for j=0:14:columnas1

        aa3 = 3+i;
        a3(1,r3) = aa3;

        c3 = 6+i;
        k3(1,r3) = c3;

        bb3 = 2+j;
        b3(1,r3) = bb3;

        d3 = 9+j;
        l3(1,r3) = d3;

        r3 = r3+1;
    end
end

a33 = zeros(length(a2));
a33(1:length(a3)) = a3;
k33 = zeros(length(k2));
k33(1:length(k3)) = k3;
b33 = zeros(length(b2));
b33(1:length(b3)) = b3;
l33 = zeros(length(l2));
l33(1:length(l3)) = l3;

% Definimos las nuevas posiciones no usadas (Antena4)

a4 = zeros(1,ceil(filas1/6));
k4 = zeros(1,ceil(filas1/6));
b4 = zeros(1,ceil(columnas1/14));
l4 = zeros(1,ceil(columnas1/14));
r4 = 1;

for i=(GP+0):6:(filas1+0-GP)
    for j=0:14:columnas1

        aa4 = 6+i;
        a4(1,r4) = aa4;

        c4 = 3+i;
        k4(1,r4) = c4;

        bb4 = 2+j;
        b4(1,r4) = bb4;

        d4 = 9+j;
        l4(1,r4) = d4;

        r4 = r4+1;
    end
end

a44 = zeros(length(a2));
a44(1:length(a4)) = a4;
k44 = zeros(length(k2));
k44(1:length(k4)) = k4;
```

```
b44 = zeros(length(b2));
b44(1:length(b4)) = b4;
l44 = zeros(length(l2));
l44(1:length(l4)) = l4;

for p=1:columnas1
    for o=(GP+1):(filas1-GP)

        for m=1:length(k1)
            if ((k1(m)==o && l1(m)==p))
                salidaSPtx(o,p)=0;
                break;
            elseif (a1(m)==o && b1(m)==p)
                salidaSPtx(o,p)=0;
                break;
            elseif ((k2(m)==o && l2(m)==p))
                salidaSPtx(o,p)=0;
                break;
            elseif (a2(m)==o && b2(m)==p)
                salidaSPtx(o,p)=0;
                break;
            elseif (a33(m)==o && b33(m)==p)
                salidaSPtx(o,p)=P;
                break;
            elseif (k33(m)==o && l33(m)==p)
                salidaSPtx(o,p)=P;
                break;
            elseif (a44(m)==o && b44(m)==p)
                salidaSPtx(o,p)=0;
                break;
            elseif (k44(m)==o && l44(m)==p)
                salidaSPtx(o,p)=0;
                break;
            else
                salidaSPtx(o,p)=salidaSPtx(o,p);
            end
        end
    end
end
end

--------------------------------------------------------------------------------

function [H] = ecualizadorMatrizEnteraMIMOAntena3 (salidafft,P,GP)

[filas,columnas] = size(salidafft);
h = zeros(filas-2*GP,columnas);
% Número pilotos por columna
NP = floor(2*(filas-2*GP)/12);

    for j=0:14:(columnas-14)
        % 2ª columna
        a = zeros(NP,1);
        posicion1 = 6;
        for i=1:NP
            if ((GP+posicion1)<(filas-GP+1))
                a(i,1)= salidafft(GP+posicion1,j+2);
                posicion1 = posicion1+6;
            else
                break;
            end
        end
        aifft = ifft(a);
        aifft = [aifft ;zeros((filas-NP-2*GP),1)];
        afft = fft(aifft);
        h(:,j+2) = afft;
```

```
        %9ª columna
        b = zeros(NP,1);
        posicion5 = 3;
        for k=1:NP
            if ((GP+posicion5)<(filas-GP+1))
                b(k,1)= salidafft(GP+posicion5,j+9);
                posicion5 = posicion5+6;
            else
                break;
            end
        end
        bifft = ifft(b);
        bifft = [bifft ;zeros((filas-NP-2*GP),1)];
        bfft = fft(bifft);
        h(:,j+9) = bfft;
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%
% Estimación en la dimensión temporal

        % INTERPOLACIONES
        for n=1:(filas-2*GP)
            for j=0:14:(columnas-14)

            % Interpolación 2ª y 9ª columna
            a1 = h(n,2+j);
            b1 = h(n,9+j);
            x1 = [1 8]; %pos. conocidas
            y1real = [real(a1) real(b1)];%valores de dichas posiciones
            z1real = interp1(x1,y1real,[2 3 4 5 6 7]);
            y1imag = [imag(a1) imag(b1)];
            z1imag = interp1(x1,y1imag,[2 3 4 5 6 7]);
            z21 = z1real(1) +1i*z1imag(1);
            z31 = z1real(2) +1i*z1imag(2);
            z41 = z1real(3) +1i*z1imag(3);
            z51 = z1real(4) +1i*z1imag(4);
            z61 = z1real(5) +1i*z1imag(5);
            z71 = z1real(6) +1i*z1imag(6);
            h(n,2+j) = z21;
            h(n,3+j) = z31;
            h(n,4+j) = z41;
            h(n,5+j) = z51;
            h(n,6+j) = z61;
            h(n,7+j) = z71;

            % Interpolación 9ª y 16ª columna
            if (j==(columnas-14))
                break;
            else
            a2 = h(n,9+j);
            b2 = h(n,16+j);
            x2 = [1 8];%pos. conocidas
            y2real = [real(a2) real(b2)];%valores de dichas posiciones
            z2real = interp1(x2,y2real,[2 3 4 5 6 7]);
            y2imag = [imag(a2) imag(b2)];
            z2imag = interp1(x2,y2imag,[2 3 4 5 6 7]);
            z22 = z2real(1) +1i*z2imag(1);
            z32 = z2real(2) +1i*z2imag(2);
            z42 = z2real(3) +1i*z2imag(3);
            z52 = z2real(4) +1i*z2imag(4);
            z62 = z2real(5) +1i*z2imag(5);
            z72 = z2real(6) +1i*z2imag(6);
            h(n,10+j) = z22;
            h(n,11+j) = z32;
            h(n,12+j) = z42;
```

```
            h(n,13+j) = z52;
            h(n,14+j) = z62;
            h(n,15+j) = z72;
            end

            end
        end

        h(:,1) = h(:,2);
        h(:,columnas-4) = h(:,columnas-5);
        h(:,columnas-3) = h(:,columnas-5);
        h(:,columnas-2) = h(:,columnas-5);
        h(:,columnas-1) = h(:,columnas-5);
        h(:,columnas) = h(:,columnas-5);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%

H1 = zeros(size(h));
[filas1 columnas1] = size(H1);
a = 4;
for i=1:(filas1-a)
    H1(i+a,:) = h(i,:);
end
for j=(filas1-a):filas1
    H1((j-filas1+a+1),:) = h(j,:);
end

h1 = zeros(GP,columnas);
H = [h1;H1;h1];

end

-------------------------------------------------------------------------------

function [H] = ecualizadorMatrizEnteraMIMOAntena4 (salidafft,P,GP)

[filas,columnas] = size(salidafft);
h = zeros(filas-2*GP,columnas);
% Número pilotos por columna
NP = floor(2*(filas-2*GP)/12);

    for j=0:14:(columnas-14)
        % 2ª columna
        a = zeros(NP,1);
        posicion1 = 3;
        for i=1:NP
            if ((GP+posicion1)<(filas-GP+1))
                a(i,1)= salidafft(GP+posicion1,j+2);
                posicion1 = posicion1+6;
            else
                break;
            end
        end
        aifft = ifft(a);
        aifft = [aifft ;zeros((filas-NP-2*GP),1)];
        afft = fft(aifft);
        h(:,j+2) = afft;

        %9ª columna
        b = zeros(NP,1);
        posicion5 = 6;
        for k=1:NP
            if ((GP+posicion5)<(filas-GP+1))
                b(k,1)= salidafft(GP+posicion5,j+9);
                posicion5 = posicion5+6;
            else
                break;
```

```
            end
        end
        bifft = ifft(b);
        bifft = [bifft ;zeros((filas-NP-2*GP),1)];
        bfft = fft(bifft);
        h(:,j+9) = bfft;
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%
% Estimación en la dimensión temporal

        % INTERPOLACIONES
        for n=1:(filas-2*GP)
            for j=0:14:(columnas-14)

            % Interpolación 2ª y 9ª columna
            a1 = h(n,2+j);
            b1 = h(n,9+j);
            x1 = [1 8]; %pos. conocidas
            y1real = [real(a1) real(b1)];%valores de dichas posiciones
            z1real = interp1(x1,y1real,[2 3 4 5 6 7]);
            y1imag = [imag(a1) imag(b1)];
            z1imag = interp1(x1,y1imag,[2 3 4 5 6 7]);
            z21 = z1real(1) +1i*z1imag(1);
            z31 = z1real(2) +1i*z1imag(2);
            z41 = z1real(3) +1i*z1imag(3);
            z51 = z1real(4) +1i*z1imag(4);
            z61 = z1real(5) +1i*z1imag(5);
            z71 = z1real(6) +1i*z1imag(6);
            h(n,2+j) = z21;
            h(n,3+j) = z31;
            h(n,4+j) = z41;
            h(n,5+j) = z51;
            h(n,6+j) = z61;
            h(n,7+j) = z71;

            % Interpolación 9ª y 16ª columna
            if (j==(columnas-14))
                break;
            else
            a2 = h(n,9+j);
            b2 = h(n,16+j);
            x2 = [1 8];%pos. conocidas
            y2real = [real(a2) real(b2)];%valores de dichas posiciones
            z2real = interp1(x2,y2real,[2 3 4 5 6 7]);
            y2imag = [imag(a2) imag(b2)];
            z2imag = interp1(x2,y2imag,[2 3 4 5 6 7]);
            z22 = z2real(1) +1i*z2imag(1);
            z32 = z2real(2) +1i*z2imag(2);
            z42 = z2real(3) +1i*z2imag(3);
            z52 = z2real(4) +1i*z2imag(4);
            z62 = z2real(5) +1i*z2imag(5);
            z72 = z2real(6) +1i*z2imag(6);
            h(n,10+j) = z22;
            h(n,11+j) = z32;
            h(n,12+j) = z42;
            h(n,13+j) = z52;
            h(n,14+j) = z62;
            h(n,15+j) = z72;
            end

            end
        end

        h(:,1) = h(:,2);
        h(:,columnas-4) = h(:,columnas-5);
```

```
        h(:,columnas-3) = h(:,columnas-5);
        h(:,columnas-2) = h(:,columnas-5);
        h(:,columnas-1) = h(:,columnas-5);
        h(:,columnas) = h(:,columnas-5);
    %%%%%%%%%%%%%%%%%%%%%%%%%%
H1 = zeros(size(h));
[filas1 columnas1] = size(H1);
a = 4;
for i=1:(filas1-a)
    H1(i+a,:) = h(i,:);
end
for j=(filas1-a):filas1
    H1((j-filas1+a+1),:) = h(j,:);
end

h1 = zeros(GP,columnas);
H = [h1;H1;h1];

end

-------------------------------------------------------------------------------

function [salidaPSrx] = conversorPSrxMIMO4antenas(salidaEcualizador0,GP)

[filas0,columnas0] = size(salidaEcualizador0);
salidaEcualizador = salidaEcualizador0((GP+1):(filas0-GP),:);
[filas,columnas] = size(salidaEcualizador);
NelementosAnadidos = (12*filas*columnas)/(12*7);
salidaPSrx = zeros(1,filas*columnas-NelementosAnadidos);

% Definimos las posiciones a quitar de la rejilla de recursos
a1 = zeros(1,ceil(filas/3));
b1 = zeros(1,ceil(columnas/7));
b2= zeros(1,ceil(columnas/7));
b3 = zeros(1,ceil(columnas/7));
p1 = 1;
for i=0:3:(filas-1)
    for j=0:7:(columnas-1)

        aa1 = 3+i;
        a1(1,p1) = aa1;

        bb1 = 1+j;
        b1(1,p1) = bb1;

        bb2 = 2+j;
        b2(1,p1) = bb2;

        bb3 = 5+j;
        b3(1,p1) = bb3;

        p1 = p1+1;
    end
end

q=1;
for n=1:columnas
   for o=1:filas
       % Recorremos la matriz con el objetivo de extraer la salida del
       % ecualizador que se trata de un vector lineal
       % Si llegamos a una posición de la matriz correspondiente a una
       % señal piloto no usamos esa posición en la salida del ecualizador, en caso
       % contrario introducimos el símbolo de tráfico del modulador en la
       % posición correspondiente de la salida del ecualizador
       for m=1:length(b1)
            if (a1(m)==o && b1(m)==n)
```

```
                q = q-1;
                break;
            elseif (a1(m)==o && b2(m)==n)
                q = q-1;
                break;
            elseif ((a1(m)==o && b3(m)==n))
                q = q-1;
                break
            else
                salidaPSrx(q)=salidaEcualizador(o,n);
            end
        end
        q=q+1;
    end
end
end
```

GLOSARIO

3GPP: Third Generation Partnership Project
2G: Segunda generación
3G: Tercera generación
4G: Cuarta generación
BS: Base Station
CDMA: Code Division Multiple Access
DAB: Digital Audio Broadcasting
DFT: Discrete Fourier Transform
DL: Downlink
DSL: Digital Subscriber Line
EDGE: Enhanced Data Rates for GSM Evolution
EIR: Equipment Identity Register
eNodoB: Evolved Nodo B
EPC: Evolved Packet Core
EPS: Evolved Packet System
E-UTRAN: Evolved UTRAN Network o Red de Acceso Radio
FDD: Frequency-division duplexing
FFT: Fast Fourier Transform
FS: Full Speed
GMSK: Gaussian minimum shift keying
GPRS: General packet radio service
GSM: Global System for Mobile Communications
HS: Half Speed
HSCSD: High-Speed Circuit-Switched Data
HSDPA: High-Speed Downlink Packet Access
HSPA: High-Speed Packet Access
HSS: Home Subscriber Server
HSUPA: High-Speed Uplink Packet Access
IDFT: Inverse Discrete Fourier Transform
IFFT: Inverse Fast Fourier Transform
IMT-Advanced: International Mobile Telecommunications-Advanced
IP: Internet Protocol
ISI: Inter Symbol Interference
LTE: Long Term Evolution
MIMO: Multiple-input Multiple-output
MISO: Multiple-input Single-output
MME: Mobile Management Entity
MS: Mobile Station
OFDM: Orthogonal Frequency Division Multiplexing
QAM: Quadrature amplitude modulation
QPSK: Quadrature Phase Shift Keying

P/S: Paralelo / Serie
PCRF: Policy and Charging Rules Function
PLC: Power Line Communications
P-GW: Packet Data Network Gateway, PDN-GW
PSK: Phase Shift Keying
RB: Resource Block
RE: Resource Element
RNC: Radio Network Controller
S/P: Serie / Paralelo
SCM: Spatial Channel Model
S-GW: Serving Gateway
SIMO: Single-input Multiple-output
SISO: Single-input Single-output
SMS: Short Message Service
SNR: Signal to Noise Ratio
TCP: Transmission Control Protocol
TDMA: Time division multiple access
UE: User Equipment
UMTS: Universal Mobile Telecommunications System
WCDMA: Wideband Code Division Multiple Access
WiMAX: Worldwide Interoperability for Microwave Access

BIBLIOGRAFÍA

[1] Apuntes de la asignatura de Comunicaciones Móviles de 4º del Grado en Ingeniería de Sistemas de Comunicaciones del curso 2012/13.

[2] GPRS y EDGE
http://www.3gpp.org/Technologies/Keywords-Acronyms/article/gprs-edge
Enlace disponible el 24 de Septiembre de 2013.

[3] HSPA+ http://www.3gpp.org/HSPA
Enlace disponible el 24 de Septiembre de 2013.

[4] Artículo de Félix García en Network World del 01/04/2009 "LTE el estándar esperado"
http://www.networkworld.es/movilidad/lte-el-estandar-esperado
Enlace disponible el 24 de Septiembre de 2013.

[5] Artículo del periódico ABC del 28/05/2013 "¿Qué falta para que llegue LTE a España?"
http://www.abc.es/tecnologia/redes/20130317/abci-falta-mucho-para-espana-201303152021.html
Enlace disponible el 24 de Septiembre de 2013.

[6] LTE http://www.3gpp.org/LTE
Enlace disponible el 24 de Septiembre de 2013.

[7] "LTE (Long Term Evolution): El siguiente nivel" Artículo de Octubre de 2010 del departamento de instrumentación de Rohde & Schwarz España.

[8] Holma, Toskala "LTE for UMTS: Evolution to LTE-Advanced"
Wiley, 2011.

[9] Dahlman, Erik "4G: LTE/LTE-Advanced for Mobile Broadband"

[10] Release 8 de 3GPP
LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation (3GPP TS 36.211 version 8.4.0 Release 8)

[11] "Desarrollo de algoritmos de estimación de canal, sincronismo y CAG para WLAN basada en OFDM" DTSC-UC3M

[12] Paulraj, Arogyaswami "Introduction to space-time wireless communications"
Cambridge UniversityPress, 2003.

[13] "Análisis de rendimiento de sistemas MIMO-SDM OFDM" Universidad de Málaga

[14] MATLAB implementation of the 3GPP Spatial Channel Model (3GPP TR 25.996)

[15] Matlab The Language of Technical Computing.
http://www.mathworks.es/products/matlab/
Enlace disponible el 24 de Septiembre de 2013.

[16] MATLAB and Simulink Student Version.
http://www.mathworks.es/academia/student_version/?s_cid=0210_desp_es_spa_16806167
Enlace disponible el 24 de Septiembre de 2013.